REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
30-06-2011	Final Report	4/1/2008 – 3/31/2011

4. TITLE AND SUBTITLE

Coordinating Learning Agents for Active Information Collection

5a. CONTRACT NUMBER
FA9550-08-1-0187

5b. GRANT NUMBER
FA9550-08-1-0187

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Kagan Tumer

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Oregon State University
204 Rogers Hall
Corvallis, OR 97331

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Airforce Office of Scientific Research

10. SPONSOR/MONITOR'S ACRONYM(S)
AFOSR

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
AFRL-OSR-VA-TR-2012-0252

12. DISTRIBUTION / AVAILABILITY STATEMENT

Publicly available

13. SUPPLEMENTARY NOTES

14. ABSTRACT
The ability to autonomously coordinate a team of agents to actively collect information is critical to a wide array of Air Force missions. With computing power becoming both cheaper and more powerful, there is a trend to push critical decision making capabilities ``downstream", towards the data collection nodes rather than wait for data to arrive to a massive centralized location before a decision is made. This new computing paradigm relies on networked agents to actively collect, process and query data and promises to significantly improve both the quality/relevance of the collected data and the associating decision making. This project provides a comprehensive solution to the problem of intelligent data gathering and decision making by ensuring that the information collected by an agent has the most ``added value'' to the full network. The key contribution of this project is to shift the focus from ``how to optimize'' to ``what to optimize'' in difficult coordination problems. The impact of this work extends to a large class of problems relevant to the Air Force including satellite communication systems, reconfigurable flight control systems, sensor networks, and intelligence gathering in hybrid networks.

15. SUBJECT TERMS
Multiagent Coordination, Net-centric Utility derivation, System reconfiguration, Distributed optimization,

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			19b. TELEPHONE NUMBER *(include area code)*

Final Report – 2011

FA9550-08-1-0187

AFOSR Systems and Software Program

Coordinating Learning Agents for Active Information Collection

PI: Kagan Tumer
Organization: Oregon State University

June 30, 2011

Contents

1 Cover Sheet

Title: Coordinating Learning Agents for Active Information Collection PI: Kagan Tumer
Institution: Oregon State University
Grant number: FA9550-08-1-0187

Status and Accomplishments: See abstract below for details
Personnel Supported:

Kagan Tumer
Ehsan Nasroullahi
Scott Proper
Max Salichon
Matt Knudson
Jack Shepherd

Reporting Period Start: 4/1/2008
Reporting Period End: 3/31/2011

Changes in Research Objectives: None
Extensions Requested/Granted: None

Honors and awards (all paper numbers refer to publications list below)
1. Nomination for best applications paper at GECCO 2010 for paper number 1
2. Nomination for best applications paper at GECCO 2010 for paper number 3
3. 1st runner up for best theoretical paper at ANNIE 2008 for paper number 11
3. 2nd runner up for best applications paper at ANNIE 2008 for paper number 10

2 Executive Summary

The ability to autonomously coordinate a team of agents to actively collect information is critical to a wide array of Air Force missions. With computing power becoming both cheaper and more powerful, there is a trend to push critical decision making capabilities "downstream", towards the data collection nodes rather than wait for data to arrive to a massive centralized location before a decision is made. This new computing paradigm relies on networked agents to actively collect, process and query data and promises to significantly improve both the quality/relevance of the collected data and the associating decision making. The technological bottlenecks for such a computing scheme stem from a lack of mathematics and algorithms to manage such systems rather than difficulties associated with building and deploying them.

2.1 Objectives

This project provides a comprehensive solution to the problem of intelligent data gathering and decision making by ensuring that the information collected by an agent has the most "added value" to the full network. The three specific objectives of this project are to:

1. Derive the system properties that quantify the alignment between local and network utilities;
2. Derive (and update) agent utilities under communication and computation restrictions that will lead to good network utilities; and
3. Derive agent utilities and learning strategies for agents in dynamic and stochastic environments with "black box" network utility functions.

2.2 Key Contributions

The key contribution of this project is to shift the focus from "how to optimize" to "what to optimize" in difficult coordination problems. The impact of this work extends to a large class of problems relevant to the Air Force including satellite communication systems, reconfigurable flight control systems, sensor networks, and intelligence gathering in hybrid networks.

We obtained significant results supporting all three objectives. In particular we have:

- Applied system characteristics to derive agent utility functions for coordinating information gathering robots. Robots using such utility functions to learn actions significantly

outperform other robots in a simulated information gathering task requiring coordination. This result directly supports objective 1.

- Developed a new and fast learning algorithm supporting multiagent coordination. These results are based on learning from "actions not taken", meaning that the agents update their estimate of potential actions' outcomes based on information gathered by other agents.This results directly supports objective 2 by improving the network utility.

- Developed agent utilities that promote team formation in multiagent systems and allow agents to achieve good values of network utility without requiring communication. This results directly supports objective 2. Papers:

- Decomposed network utilities into components to allow approximations to agent utilities that promote coordination in the presence of "black box" utility functions. This results directly supports objective 3. Papers:

- Achieved coordinated behavior in a team of heterogeneous agents aiming to achieve a high level utility. This objective directly supports all three objectives and the overall goals of the proposal.

3 Publication List

Based on the work performed on this project, we published 2 journal articles and 10 conference papers, 6 of which were in highly refereed conferences with acceptance rates below 50%. In addition one journal paper was recently submitted for review.

Article submitted based on work performed in this project:

2011:

J. Sepherd III and K. Tumer. A Hierarchical Approach to Autonomous Quadrotor Control. Submitted in 2011.

Articles published based on work performed in this project:

2010:

1. J. Sepherd III and K. Tumer. Robust Neuro-Control for A Micro Quadrotor. *Proceedings of the 2010 Genetic and Evolutionary Computation Conference*. Portland, OR, July 2010 (45% acceptance). **Nominated for best application paper award.**
2. M. Knudson and K. Tumer. Coevolution of Heterogeneous Multi-Robot Teams. *Proceedings of the 2010 Genetic and Evolutionary Computation Conference*. Portland, OR, July 2010 (45% acceptance)
3. M. Salichon and K. Tumer. A Neuro-Evolutionary Approach to Micro Aerial Vehicle Control. *Proceedings of the 2010 Genetic and Evolutionary Computation Conference*. Portland, OR, July 2010 (45% acceptance). **Nominated for best application paper award.**
4. M. Knudson and K. Tumer. Robot Coordination with Ad-hoc Team Formation AAMAS (extended abstract). *Proceedings of the 2010 Genetic and Evolutionary Computation Conference*. May 2010 (43% acceptance)
5. M. Knudson and K. Tumer. Policy Search and Policy Gradient Methods for Autonomous Navigation. *Proceedings of the 2010 Genetic and Evolutionary Computation Conference*. Portland, OR, July 2010 (45% acceptance)

2009:

6. N. Khani and K. Tumer. Learning from Actions Not Taken: A Multiagent Learning Algorithm (extended abstract).*In Proceedings of the Eighth International Joint Conference on Autonomous Agents and MultiAgent Systems.* Budapest, Hungary, May 2009. (41% acceptance).

7. K. Tumer and N. Khani. Learning from Actions Not Taken in Multiagent Systems. ***Advances in Complex Systems***, Vol 12:455–473, 2009.

8. K. Tumer and A. Agogino. Multiagent Learning for Black Box System Reward Functions. ***Advances in Complex Systems***, Vol 12:475–492, 2009.

2008:

9. M. Knudson and K. Tumer. Towards Coordinating Autonomous Robots for Exploration in Dynamic Environments. In Intelligent Engineering Systems through Articial Neural Networks, Vol 18, pp. 587-594, ASME Press, November 2008.

10. M. Knudson and K. Tumer. Neuro-Evolutionary Navigation for Resource-Limited Mobile Robots. In Intelligent Engineering Systems through Artificial Neural Networks, Vol 18, pp. 27-34, ASME Press, November 2008. **2nd runner up for best application paper award.**

11. N. Khani and K. Tumer. Fast Multiagent Learning: Cashing in on Team Knowledge. In Intelligent Engineering Systems through Artificial Neural Networks, Vol 18, pp 310, ASME Press, November 2008. **1st runner up for best theoretical development paper award.**

12. M. Salichon and K. Tumer. A Neuro-evolutionary Approach to Micro Aerial Vehicle Control. In Intelligent Engineering Systems through Artificial Neural Networks, Vol 18, pp. 11-18, ASME Press, November 2008.

4 Technical Contributions

The key technical contributions of this work were to determine how to provide utilities for individual components of a team to ensure the coordinated and efficient behavior of the full team. to that end, three key results were (i) that agents can learn from each other without explicitly trying each alternative action ; (ii) agents can derive local utilities from "blackbox" network utility functions ; and t (iii) agents that learn together can achieve tight coordination if their utilities are properly derived based on the characteristics derived in objectives 1 and 2. The attached three articles provide the key results from these two scientific contributions of this work.

4.1 Learning from Actions not Taken

K. Tumer and N. Khani. Learning from Actions Not Taken in Multiagent Systems. ***Advances in Complex Systems***, Vol 12:455–473, 2009.

4.2 Learning from Blackbox Utility Functions

K. Tumer and A. Agogino. Multiagent Learning for Black Box System Reward Functions. ***Advances in Complex Systems***, Vol 12:475–492, 2009.

4.3 Coordinating Heterogeneous Teams of Robots

M. Knudson and K. Tumer. Coevolution of Heterogeneous Multi-Robot Teams. *Proceedings of the 2010 Genetic and Evolutionary Computation Conference*. Portland, OR, July 2010.

Advances in Complex Systems, Vol. 12, Nos. 4 & 5 (2009) 455–473

World Scientific
www.worldscientific.com

LEARNING FROM ACTIONS NOT TAKEN IN MULTIAGENT SYSTEMS

KAGAN TUMER* and NEWSHA KHANI†

Oregon State University, 204 Rogers Hall,
Corvallis, Oregon 97331, USA
**kagan.tumer@oregonstate.edu*
†*khanin@onid.orst.edu*

Received 31 January 2009
Revised 11 June 2009

In large cooperative multiagent systems, coordinating the actions of the agents is critical to the overall system achieving its intended goal. Even when the agents aim to cooperate, ensuring that the agent actions lead to good system level behavior becomes increasingly difficult as systems become larger. One of the fundamental difficulties in such multiagent systems is the slow learning process where an agent not only needs to learn how to behave in a complex environment, but also needs to account for the actions of other learning agents. In this paper, we present a multiagent learning approach that significantly improves the learning speed in multiagent systems by allowing an agent to update its estimate of the rewards (e.g. value function in reinforcement learning) for all its available actions, not just the action that was taken. This approach is based on an agent estimating the counterfactual reward it would have received had it taken a particular action. Our results show that the rewards on such "actions not taken" are beneficial early in training, particularly when only particular "key" actions are used. We then present results where agent teams are leveraged to estimate those rewards. Finally, we show that the improved learning speed is critical in dynamic environments where fast learning is critical to tracking the underlying processes.

Keywords: Multiagent learning; counterfactual reward; difference reward.

1. Introduction

Learning in large multiagent systems is a critical area of research with applications ranging from robocup soccer [26, 27], to rover coordination [19], to trading agents [25, 43], to air traffic management [32]. What makes this problem particularly challenging is that the agents in the system provide a constantly changing background in which each agent needs to learn its task. As a consequence, almost by definition, all multiagent learning occurs in complex environments, where the agents need to extract the underlying reward signal from the noise of the other agents acting within the same environment.

Furthermore, typically, two learning problems are coupled where the agent needs to solve both a temporal credit assignment problem (how to assign a reward received

at the end of sequence of actions to each action) and a structural credit assignment problem (how to assign credit to a particular agent at the end of a multiagent task) [1, 15, 16, 28, 38, 41, 44]. The temporal credit assignment problem has been extensively studied [10, 16, 28, 31, 30, 39, 42], and the structural credit assignment problem has recently been investigated as well [4, 8, 11, 20, 23, 35].

Learning sequences of actions for multiagent systems has blended these two areas of research and led to key advances [6, 8, 12, 26, 40]. In these cases, the learning needs of the agents are modified to account for their presence in a larger system [2, 11, 13, 22, 35, 37]. However, though these methods have yielded tremendous advances in multiagent learning, they are principally based on an agent trying an action, receiving an evaluation of that action, and updating its own estimate on the "value" of taking that action in that state. Though effective, such an approach is generally slow to converge, particularly in large and dynamic environments.

In this paper, we explore the concept of agents learning from actions they do not take by estimating the rewards they would have received had they taken those actions. These counterfactual rewards are estimated using the theory developed for structural credit assignment, and prove effective in the congestion games. Furthermore, a team structure can be used to provide the required information for the agents to compute these reward estimates [24, 29]. A key benefit of this approach is that an increase in the number of agents can be leveraged to improve the estimates of actions not taken, turning a potential pitfall (e.g. how to extract useful information from the actions of so many agents) into an asset (e.g. learn from the experiences of other agents). Though the concept of updating rewards for actions not taken is present in learning automata literature, where for example, the probability of taking a particular action may go up down based on similar actions' results [21, 38, 39], in this work we explicitly aim to quantify the counterfactual concept of "*what would my reward have been, had I taken another action.*"

In Sec. 2, we discuss the congestion problem that we use in the reported experiments. In Sec. 3, we summarize the basic agent learning architecture. In Sec. 4, we provide the action-not-taken (ANT) rewards and modify them using team rewards. We also provide experimental results showing the basic behavior of the ANT reward. In Sec. 5, we explore the application of these rewards to dynamic domains where the rapidly changing conditions put a premium on learning quickly. Finally, in Sec. 6, we discuss the results and provide directions for future research.

2. Congestion Problems

Congestion problems where system performance depends on the number of agents taking a particular action provide an interesting domain to study the behavior of cooperative multiagent systems. In congestion problems, agents need to learn how to synchronize (or not synchronize) their actions, rather than learn to take particular actions. This type of problem is ubiquitous in routing domains (e.g. on

a highway, a particular lane is not preferable to any other lane, but what matters is how many others are using a particular lane) [18, 34].

The multi-night bar problem is an abstraction of congestion games (and a variant of the El Farol bar problem [5]) which have been extensively studied [1, 5, 9, 7, 14]. In this version of the congestion problem, each agent has to determine which day in the week to attend a bar. The problem is set up so that if either too few agents attend (boring evening) or too many people attend (crowded evening), the total enjoyment of the attending agents drop.

The system performance is quantified by a system reward function G. This reward is a function of the full system state z (e.g. the joint action of all agents in the system), and is given by:

$$G(z) = \sum_{\text{day}=1}^{n} x_{\text{day}} e^{\frac{-x_{\text{day}}}{C}}, \tag{1}$$

where n is the number of actions (for example $n = 7$ if actions are days); x_{day}: the total attendance on a particular day; and C: a real-valued parameter that represents the capacity of the resource (e.g. the capacity of the bar).

What is interesting about this game is that selfish behavior by the agents tends to lead the system to undesirable states. For example, if all agents predict an empty bar, they will all attend (poor reward) or if they all predict a crowded bar, none will attend (poor reward). This aspect of the bar problem is what makes this a "congestion game" and an abstract model of many real-world problems ranging from lane selection in traffic to job scheduling across servers to data routing.

3. Basic Agent Learning

The agent actions in this problem is to select a resource (day on which to attend the bar). The learning algorithm for each agent is a simple reinforcement learner (action value). Each agent keeps an n-dimensional vector providing its estimates of the reward it would receive for taking each possible action. The system dynamics are given by:

Initialize: week 0
Repeat until week > Max week

1. agents choose actions;
2. agents' joint action leads to an overall system state;
3. the system state results in a system reward;
4. each agent receives a reward;
5. each agent updates its action selection procedure (i.e. learning);
6. week $\leftarrow$ week $+ 1$.

In any week, an agent estimates its expected reward for attending a specific night based on action values it has developed in previous weeks. At the beginning of each

training run, each agent has an equal probability of choosing each action in the first week, resulting in a uniformly random distribution across actions. At the beginning of each training week, each agent picks a night to attend based on sampling this probability vector using a Gibbs distribution. Each agent has n actions and a value V_k associated with each action a_k:

$$P_k = \frac{e^{(V_k \cdot \tau)}}{\sum_{\text{agent}} e^{(V_k \cdot \tau)}}, \tag{2}$$

where τ is a temperature term that determines the amount of exploration (low values of τ mean most actions have similar probabilities of being selected, whereas high values of τ increase the probability that the best action will be selected). Each agent receives reward R and updates the action value vector using a value function V_k:

$$V_k = (1 - \alpha) \cdot V_k + \alpha \cdot R. \tag{3}$$

A reasonable option is to provide each agent with the *full system reward* for each week. This leads to each agent receiving the reward given in Eq. (1), and using that reward to update its value estimates for each action. However, this reward is not particularly sensitive to an agent's actions and especially in large systems, leads to particularly slow learning. As a consequence, in this work, we use the *difference reward* as a starting point for the reward an agent receives after each step. Earlier work has shown that the difference reward significantly outperforms both agents receiving a purely local reward and all agents receiving the same system reward [3, 2, 33, 32, 36]. The difference reward is given by:

$$D^i(z) = G(z) - G(z - z_i), \tag{4}$$

where $z - z_i$ specifies the state of the system without agent i.[a] In this instance z is the full attendance profile of the agents, and $z - z_i$ is the attendance profile of all the agents without agent i. Difference rewards are *aligned* with the system reward, in that any action that improves the difference reward will also improve the system reward. This is because the second term on the right-hand side of Eq. (4) does not depend on agent i's actions, meaning any impact agent i has on the difference reward is through the first term (G) [32, 35]. Furthermore, it is more sensitive to the actions of agent i, reflected in the second term of D, which removes the effects of other agents (i.e. noise) from agent i's reward function.

Intuitively, this causes the second term of the difference reward function to evaluate the performance of the system without i, and therefore D measures the agent's contribution to the system reward directly. For the difference reward in the congestion problem, this amounts to having each agent estimate the system reward it would receive were it to take or not take a particular action. In this work, agents

[a]In this paper, we will use zero padded vector addition and subtraction to specify the state dependence on specific components of the system.

do not explicitly communicate with one another, and therefore, the only effect each agent has on the system is to increase the attendance, x_{day}, for night k by 1. This leads to the following difference reward:

$$\begin{aligned} D^i(z) &= G(z) - G(z - z_i) \\ &= x_{\text{day}_i} e^{\frac{-x_{\text{day}_i}}{C}} - (x_{\text{day}_i} - 1) e^{\frac{-(x_{\text{day}_i}-1)}{C}}, \end{aligned} \tag{5}$$

where x_{day_i} is the total attendance on the day selected by agent i.

4. Action-Not-Taken (ANT) Rewards

Though the difference reward given in Eq. (5), provides a reward tuned to an agent's actions, it is still based on an agent sampling each of its actions a (potentially large) number of times. In this work, in order to increase the learning speed, we introduce the concept of ANT rewards.[17] The goal with ANT rewards is to provide estimates of how the system would have turned out had an agent taken a particular action. The mathematics that allow the computation of the difference reward can be used to compute this type of reward.

In this paper, rather than have a separate results section, we provide experimental results directly alongside the reward descriptions to motivate the improvements to the rewards and the derivation of new rewards. All results are based on 20 independent runs with the standard error plotted when large enough to be relevant. Unless otherwise specified (as with the scaling runs or congestion dependent runs) the number of agents in the system was set to 120, with $C = 6$ (capacity), and $n = 5$ (number of actions, or days).

4.1. *Basic action-not-taken reward*

The direct application of this concept is to have agents update their reward estimate based on the reward they would have received had they taken other actions. Therefore, at each time step, agents perform a mathematical operation that simulates their taking a different action and compute the counterfactual reward that would have resulted from that action. For an agent i who selected action a at this step, the counterfactual reward for action b is given by:

$$D^{i \to b}(z) = G(z - z_i^a + z_i^b) - G(z - z_i^a), \tag{6}$$

where $D^{i \to b}$ is the reward for agent i taking action b; z_i^a is the state component where agent i has taken action a; z_i^b is the state component where agent i has taken action b.

The second term of Eq. (6) ($G(z - z_i^a)$) is the same as the second term of Eq. (4). Namely the reward for the state where agent i has not taken the particular action that it took. The first term though is the key to the ANT reward. In this case, we compute the reward that would have resulted had agent i taken action b rather than action a.

Utilizing this structure, D^i_{ANT} can then be formulated as shown in Eq. (7):

$$D^i_{\text{ANT}} = \begin{cases} G(z) - G(z - z^a_i), & \text{for } i \to a, \\ G(z - z^a_i + z^b_i) - G(z - z^a_i), & \text{for } i \to b \neq a, \end{cases} \tag{7}$$

where $i \to a$ means that agent i has taken action a. Note, the removal of the state in which agent i has taken action a in the second term represents the system state without agent i. Because agent i had taken action a, this removal results in a state where agent i has taken neither action a nor action b (which it has never taken). Hence the second term is the same for both conditions of Eq. (7).

Figure 1 shows the learning curves for D and D_{ANT} along with results where agents directly use the system reward G and a local reward L to learn. The local reward L is based on the agents simply receiving the reward for the action they took, which in this instance is the component of Eq. (1) corresponding to the day they attended the bar ($L = x_{\text{day}} e^{\frac{-x_{\text{day}}}{C}}$). This is a "selfish" reward, in that the agent is only concerned with the day on which it decided to attend. Though yielding poor results in this case, this is the naive decomposition of G to its components [45].

In all the experiments, the system performance is measured with respect to G, regardless of how the agents were trained. As previously noted, these results confirm that agents using D significantly outperform agents using G or L in this domain. G learns little, and L learns to do the wrong thing: Because the agent rewards are not aligned, agents aiming to maximize their own reward lead to poor system states. We include the results for agents using G and L here for completeness, but we will omit them in subsequent figures.

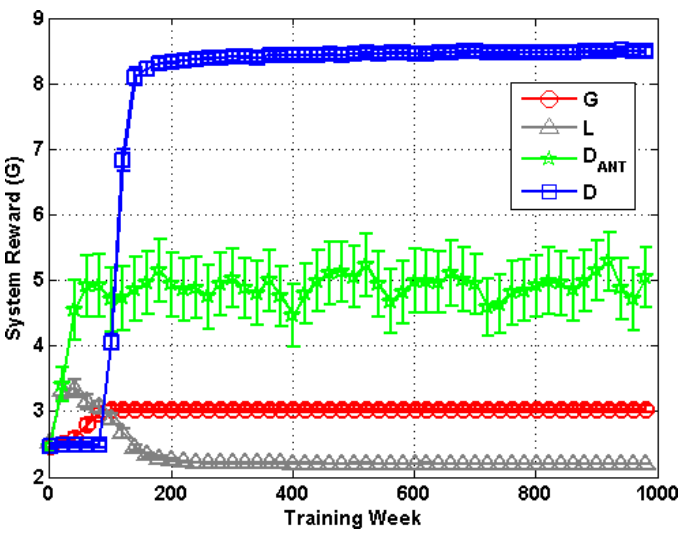

Fig. 1. System performance versus training weeks. In comparison to D, D_{ANT} based on actions not taken learns faster but shows a lower and noisier performance.

The results here show that although D_{ANT} learns faster than D, it struggles to reach good solutions. This shows that the ANT reward has a difficult time estimating the reward for most actions once those actions have been sampled. Even though the agents take advantage of such rewards and learn faster in the first weeks of training, there is a time after which these additional rewards become detrimental to the learning process. This suggests two possible solutions, which we explore in the next two sections:

(1) Use the ANT reward early in the process, but stop and switch to basic D after a "stop week."
(2) Select only a subset of the actions to receive the ANT reward.

4.2. *ANT reward with early stopping*

First, let us consider the early stopping concept to mitigate the noisy feedback agents receive for their actions. This modification is based on the observation that the ANT rewards are better than random rewards, but not as good as rewards that have been updated by actually taking the actions. Figure 2 shows the impact of having agents use ANT rewards for the first 6 weeks and then switch back to using D (the impact of when to stop is discussed in Fig. 3). Results show that this approach significantly speeds up the learning process, though does not result in agents reaching higher performance.

Figure 3 shows the dependence of the system performance on the length of time the ANT reward is used. The learning speed is stable for small values of the stop week, but starts to drop slowly as the actions not taken are used more extensively. There is a steady rightward shift as the stop week moves from 6 to 100, at which

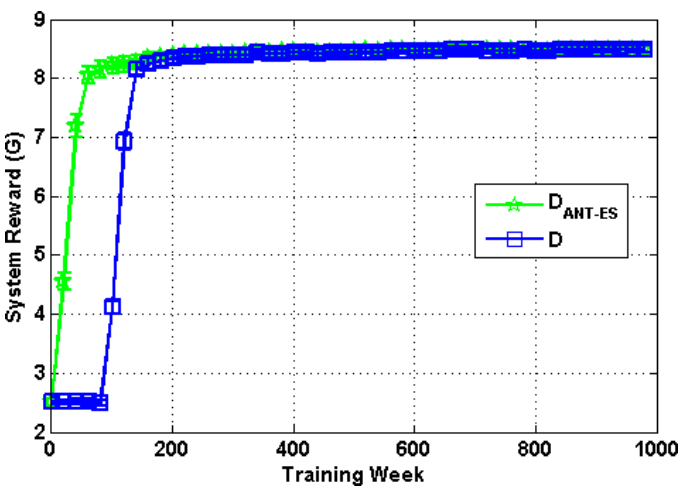

Fig. 2. System performance when actions not taken are stopped after week 6. $D_{\text{ANT-ES}}$ (ANT with Early Stopping) learns faster and reaches the same system rewards as D.

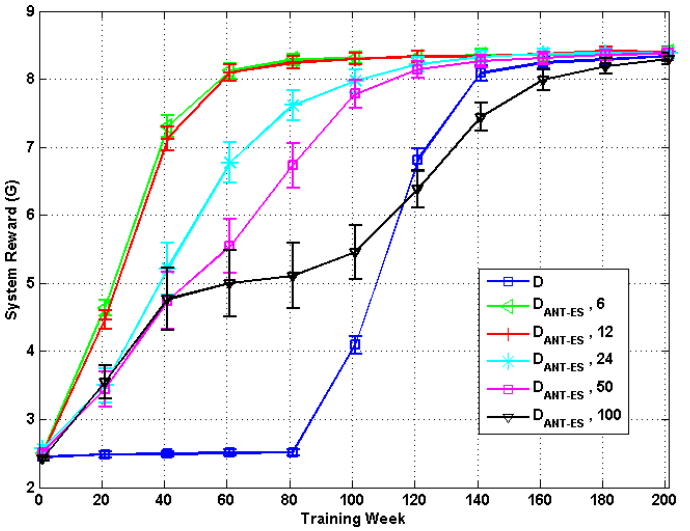

Fig. 3. The impact of the stop week on system performance. The learning speed is directly related to the length of time the action-not-taken reward is used.

point, the system learns more slowly than D alone. Providing a mechanism for selecting the stop week based on either a preset number of ANT rewards, or given performance criteria would provide automation, though in this work, we simply base the stop week on trial and error based on Fig. 3.

4.3. *ANT reward with teams*

The second option we consider is to limit the actions that are updated based on counterfactual rewards to reliable actions sampled by a subset of agents. To that end, we introduce the concept of a team, and denote agent i's team members by T_i. In this context, T_i is a fixed, randomly selected subset of the agents. This formulation gives:

$$D^i_{\text{ANT-}L} = \begin{cases} G(z) - G(z - z^a_i), & \text{for } i \to a, \\ G(z - z^a_i + z^b_i) - G(z - z^a_i), & \text{for } i \to b \in T_i, \\ 0, & \text{otherwise,} \end{cases} \tag{8}$$

where $i \to b \in T_i$ means agent i selects actions b that are sampled by agent's i's teammates T_i. As previously, the removal of agent i in the second term represents the system state without agent i having taken either action a (which it had taken) or action b (which it had not taken), leading to the term being the same in both cases.

Figure 4 shows the results when an agent has 12 randomly selected team members (in this case there are 120 total agents, so the team sizes are 10% of the total agents). Other than at the extremes (e.g. team size of 2 or 110), the experiments were not particularly sensitive to this parameter. By limiting the number of actions that are updated ($D_{\text{ANT-}L}$ in black/dark), the variability of the reward is reduced as

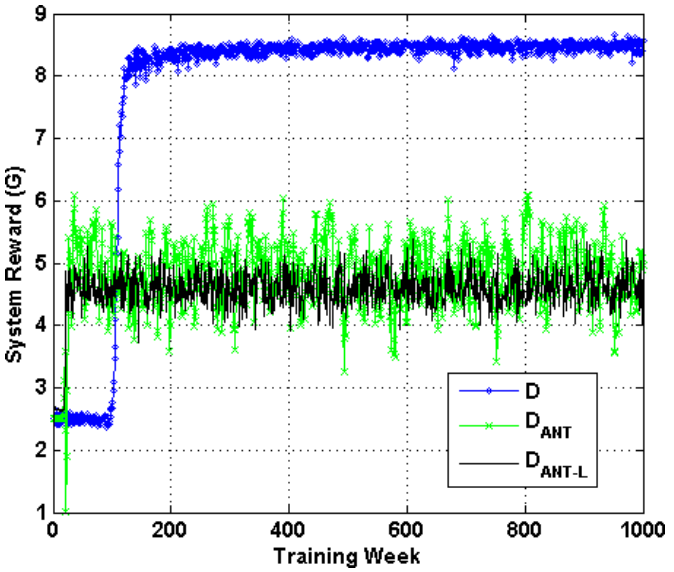

Fig. 4. System performance when only a subset of actions are explored by an agent (120 total agents, team size of 12). D performs well. $D_{\text{ANT-}L}$ based on a limited number of actions not taken performs similarly as D_{ANT} but shows a response with a lower noise level.

compared to the full D_{ANT} (in green/light), but there is no discernible improvement in the quality of the solution. However, from a computational and communication perspective, this is an interesting result, which points to a significant reduction in the need for counterfactual reward computation without loss of convergence speed.

We now combine the two concepts and have agents use teams and early stopping. Furthermore, instead of using the team members as information sources only, we increase the connection among team members by providing them all with the same reward. That is, all team members attending a particular day will receive the same reward. The learning strategy is to use team information only during the first weeks (three in the reported results, but the performance is similar for minor changes to this parameter) of learning and switch to the regular difference reward [Eq. (5)] for the rest of the training period.

The key aspect of this approach is that the team members measure the impact of a team not taking a particular action, rather than an individual agent. As a result, agents learn with their team in a smaller state space defined by the world minus their team space instead of the entire world. This is conceptually similar to the reward described in Eq. (8) but where the impact of the whole team, rather than agent i is removed, leading to:

$$D^i_{\text{Team}} = \begin{cases} G(z) - G(z - z^a_{T_i}), & \text{for } T_i \to a, \\ G(z - z^a_i + z^b_j) - G(z - z^b_{T_i} - z^a_i), & \text{for } i \to b \in T^i, \\ 0, & \text{otherwise,} \end{cases} \tag{9}$$

where $z^a_{T_i}$ is the state component of team members of agent i taking action a. In this formulation, the impact of all of agent's i teammates are removed before the

reward is calculated. Note in this case, unlike in Eqs. (7) and (8), the second term is different for the two actions. This is because this term estimates the impact of removing all team members of i that had taken a particular action. When agent i changes its action, this also changes the team members taking the same action as i. For the action a selected by agent i, we only need to remove all its team members who took that action. But to find the counterfactual reward for action b, we need to remove the actual action of agent i (action a) and then remove the team members who had taken action b. Though conceptually similar to previous rewards, the presence of team members leads to this subtle difference in the computation of the team ANT reward.

Now, let us explicitly compute D^i_{Team} for the congestion problem considered in this paper. First, for the action taken by agent i [first line of Eq. (9)], the reward becomes:

$$\begin{aligned} D^{i\to a}_{\text{Team}} &= G(z) - G(z - z^a_{T_i}) \\ &= \sum_{\text{day}} x_{\text{day}} e^{\frac{-x_{\text{day}}}{C}} - \Bigg(\sum_{\text{day}\neq \text{day}_i} x_{\text{day}} e^{\frac{-x_{\text{day}}}{C}} \\ &\quad + (x_{\text{day}_i} - |T^i_{\text{day}_i}|) e^{\frac{-(x_{\text{day}_i} - |T^i_{\text{day}_i}|)}{C}} \Bigg), \end{aligned} \tag{10}$$

where $z - z^a_{T_i}$ is the state component in which agent i and its teammate taking action a have no effect; day_i is the day agent i selects to attend; x_{day_i} is the attendance on the day agent i selects to attend; and $|T^i_{\text{day}_i}|$ is the number of agent i's teammates that choose day_i to attend.

Second, let us focus on the actions not taken by agent i [second line of Eq. (9)]. This is the reward agent i would have received had it taken the actions b chosen by some of its teammates, leading to:

$$\begin{aligned} D^{i\to b}_{\text{Team}} &= G(z - z^a_i + z^b_i) - G(z - z^b_{T_i} - z^a_i) \\ &= \sum_{\text{day}\neq \text{day}_{i\to a,b}}^{\text{day}} x_{\text{day}} e^{\frac{-x_{\text{day}}}{C}} + (x_{\text{day}_{i\to a}} - 1) e^{\frac{-(x_{\text{day}_{i\to a}} - 1)}{C}} \\ &\quad + (x_{\text{day}_{i\to b}} + 1) e^{\frac{-(x_{\text{day}_{i\to b}} + 1)}{C}} \\ &\quad - \Bigg(\sum_{\text{day}\neq \text{day}_{i\to a,b}} x_{\text{day}} e^{\frac{-x_{\text{day}}}{C}} + (x_{\text{day}_{i\to a}} - 1) e^{\frac{-(x_{\text{day}_{i\to a}} - 1)}{C}} \\ &\quad + (x_{\text{day}_{i\to b}} - |T^i_{\text{day}_{i\to b}}|) \cdot e^{\frac{-(x_{\text{day}_{i\to b}} - |T^i_{\text{day}_{i\to b}}|)}{C}} \Bigg) \\ &= (x_{\text{day}_{i\to b}} + 1) e^{\frac{-(x_{\text{day}_{i\to b}} + 1)}{C}} + (x_{\text{day}_{i\to b}} - |T^i_{\text{day}_{i\to b}}|) \cdot e^{\frac{-(x_{\text{day}_{i\to b}} - |T^i_{\text{day}_{i\to b}}|)}{C}} \end{aligned} \tag{11}$$

where $z - z_i^a + z_i^b$ is the state component in which agent i takes action b rather than action a; $z - z_{T_i}^b - z_i^a$ is the state component on which agent i (taking action a) and its teammates taking action b are removed from the state; $x_{i \to b}$ is the attendance resulting from agent i taking action b; $|T^i_{\text{day}_{i \to b}}|$ is the number of agent i's teammates that choose to attend on day resulting from action b.

In this formulation, if the agent i's team members have taken all the possible actions, each action that agent i had not taken will still be updated. Otherwise, only actions taken by i's teammates will be available for reward information and therefore updated.

Figure 5 shows the learning curves for D , $D_{\text{ANT-ES}}$ and D_{TEAM}. Agents using D_{TEAM} not only learn faster, but also reach higher system rewards than agents using the baseline D or previous variants of D_{ANT}. In this instance, not only information from team members was used, but also the reward of each team member was the same, resulting in a larger "block" of agents receiving a reward, and removing a significant amount of noise from the rewards.

4.4. *ANT reward with weighted teams*

The use of team rewards provided tangible benefits, though it treated all information received from team members equally. Yet, one can consider that the more team members take a particular action, the more reliable the estimate for the reward of

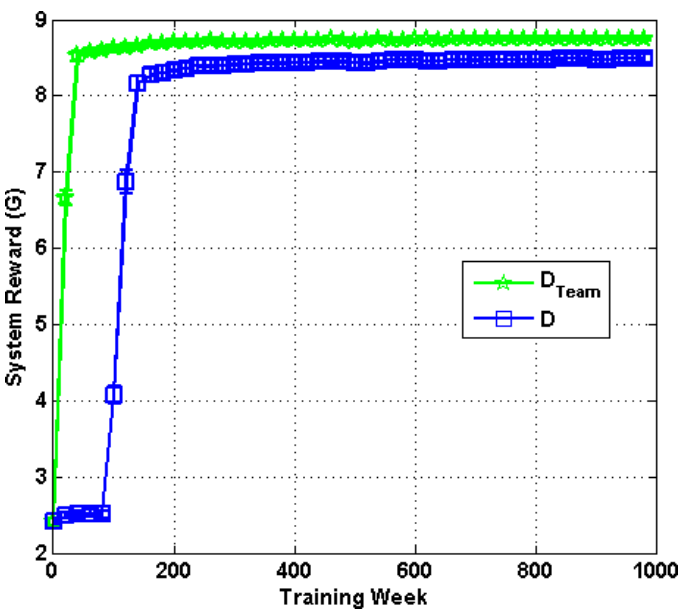

Fig. 5. System performance versus training weeks. D performs well, but D_{Team} based on updating only actions that were taken by team members both learns faster and reaches higher system rewards than D or $D_{\text{ANT-ES}}$.

that action would become. This becomes particularly relevant when the congestion in the system increases.

A simple solution to this problem is to use a weighting factor for the second term of the counterfactual reward function. In this work, we use the average number of team members selecting particular actions, though more sophisticated methods can also be used. This leads to modifying Eq. (9), that for agent i and action b leads to a weighted team reward D_{WT}:

$$D_{\mathrm{WT}}^{i\to b} = G(z - z_i^a + z_i^b) - \mu_{|T^i_{\mathrm{day}_{i\to b}}|} \cdot G(z - z^b_{T_i} - z_i^a), \tag{12}$$

where $\mu_{|T^i_{\mathrm{day}_{i\to b}}|}$ is the average number of team members taking action b.

Figure 6 explores this idea for 460 agents in a system with seven actions and a capacity of 4. Because the optimal capacity in this case is $7 \times 4 = 28$, this creates significant congestion. The results show that traditional D starts to suffer in this case, and that the weighted D_{WT} outperforms D_{TEAM}. Figure 7 shows the impact of congestion directly as the number of agents in the system increases from 120 to 460. D_{WT} handles the increased congestion better than either D_{TEAM} or D.

5. Tracking Dynamic Environments

One of the key advantages to learning rapidly is the ability to adapt to dynamic environments where the conditions may change faster than a traditional learner can adapt. In this section, we test the performance of the ANT rewards with weighted team reward on two types of dynamic environments. First, we explore seemingly

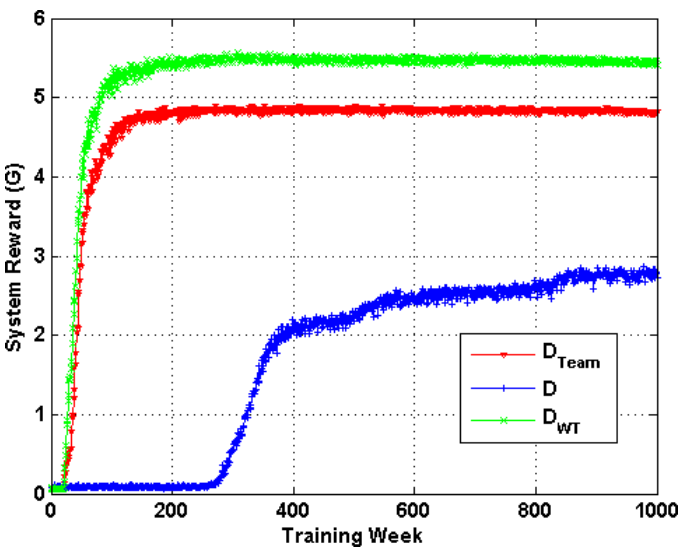

Fig. 6. System performance for the weighted team rewards. There are 460 agents in the system with only seven actions of capacity 4 leading to significant congestion. The performance of D_{WT} is significantly higher than either the base D or D_{TEAM}.

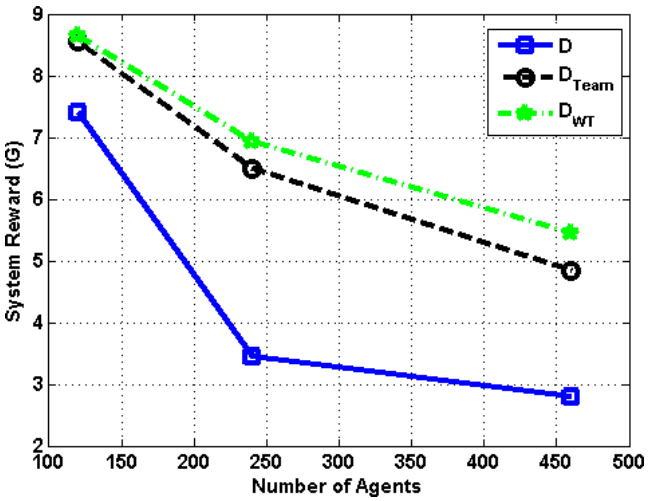

Fig. 7. The impact of congestion on system performance for the weighted team rewards. The number of agents increases, but the capacity of each day stays the same ($C = 4$). The performances of both D_{TEAM} and D_{WT} are significantly higher than D, and D_{WT} handles the congestion the best.

random changes in agent numbers and capacities, and then we explore faster, but periodic changes of both types.

5.1. *Unpredictable changes to the environment*

In this section, we explore the ability of D_{WT} to adjust to unexpected changes in the system. Figure 8 shows the system response to changes in the number of agents. In this case, the number of agents changed every 40 weeks from 280, to 140, to 180, to 100. D_{WT} not only recovers rapidly, but also learns to exploit the new condition, as demonstrated at week 120: after the initial drop caused by the change, agents using D return to their previous state, but agents using D_{WT} reach a higher system reward value.

Figure 9 shows the system response to the capacity changing from 3 to 7 every 70 weeks. D_{WT} learns faster early on and reaches slightly higher performance, but this experiment shows that D can track slow changes in the environment.

5.2. *Periodic changes to the environment*

In this section, we explore periodic and rapid changes to the environment. Figure 10 explores the performance of D_{WT} versus difference reward D when the number of agents is changing rapidly. Unlike in the results of the previous section (Fig. 8), D has a hard time tracking these changes. D_{WT} on the other hand converges to a good solution despite the number of agents in the system changing the optimal solutions for each agent. (For this experiment, we modified the value update function to account for the periodicity of the system, and allowed the value update to

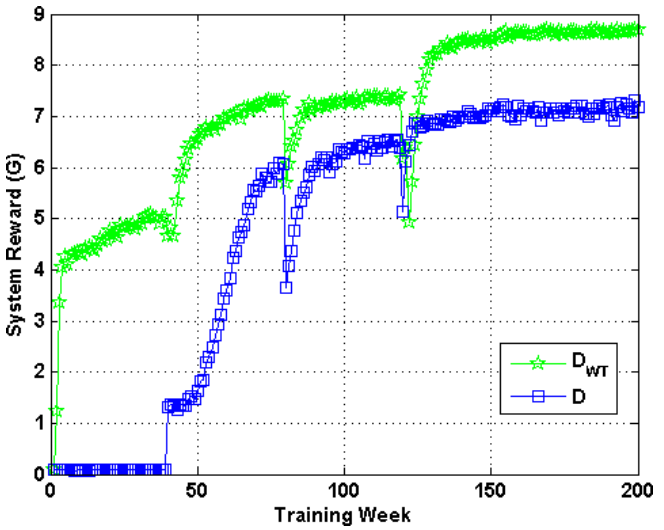

Fig. 8. System performance when the number of agents in the system changed from 280, 140, 180, 100 each 40 time steps, for seven actions and a capacity of 4. D_{WT} outperforms D both in response time and final solution quality.

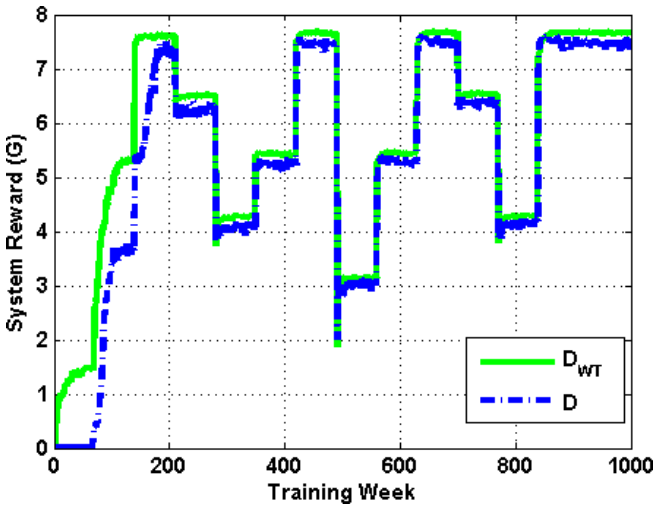

Fig. 9. System performance when the capacity of the system changes from 3 to 7 and back every 70 time steps for four actions and 120 agents.

be: $V_k = (1-\alpha)\cdot(\tau \cdot V_k^{t-1} + (1-\tau)\cdot V_k^{t'}) + \alpha \cdot R$ where t' corresponds to the last time in which the capacity was the same. This value can be estimated in practice, though in this instance, in order to remove the impact of such estimation on the reward analysis, we provided it to both reward functions.)

Finally, we explore the impact of rapid changes to the system capacity. Figure 11 shows the system performance when the capacity oscillates between 2 and 5. Unlike

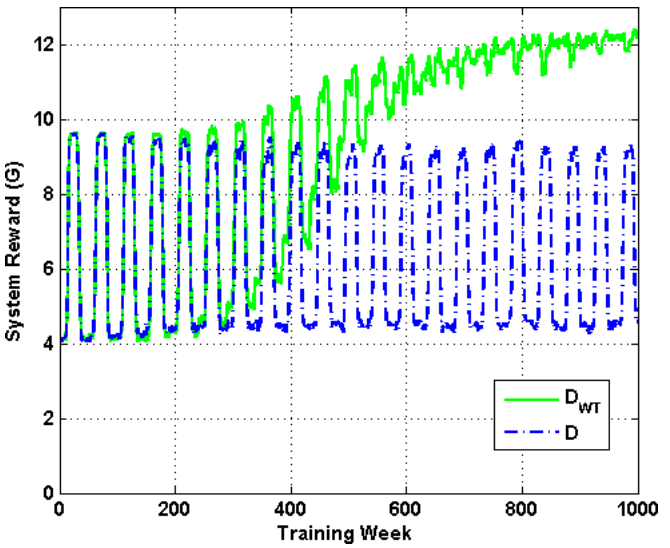

Fig. 10. System performance versus training weeks. There were eight actions with a capacity of 5. The standard difference reward D is plotted D_{WT} with variations of 60–120 in the number of agents. D cannot converge to a good solution, but D_{WT} not only converges to a good solution but does so rapidly after each capacity change.

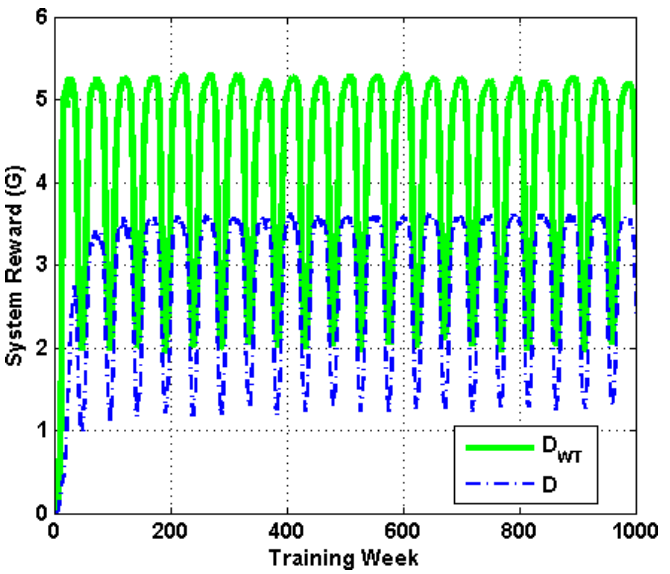

Fig. 11. System performance when the number of agents changes periodically. There were eight actions and 120 agents and capacity changed from 2 to 5 every 50 weeks. D performs poorly, but D_{WT} learns faster and reaches higher system rewards than D for both capacities.

in Fig. 9, D cannot track this continuous change as it does not get sufficient time to learn the system before the environment changes. D_{WT}, however, tracks the changes. Even though it has difficulties with the rapid changes, it both reaches higher system level performance for both $C = 2$ and $C = 5$.

6. Discussion

In large multiagent systems, the agents face a difficult learning problem where their actions are filtered through the "group action" before leading to a reward. As a consequence, an agent has a lengthy learning period where the actions need to be sampled a large number of times to extract the "signal" from the "noise." The use of the difference reward provides an improvement over directly using the system reward. However, a standard difference reward function still relies on each action being sampled before the cleaned up reward can be obtained. In this work, we present a modification to previously used difference reward, called ANT reward that provides agents with rewards on actions that were not taken by the agent.

We then provide modified versions of the ANT reward that through early stopping and team structures provides improvements in both the learning speed and the quality of the solution reached. The increase in speed of learning is the direct result of an agent receiving a counterfactual reward that estimates the reward that agent would have received had it taken a particular action. Furthermore, we show that the performance improvements are significantly more pronounced in dynamic environments where the conditions change either randomly or with high periodicity. In both cases, the rapid learning allows the agents to track a highly dynamic environment.

Though these results are encouraging, there are multiple areas for further investigation in this domain. First, the communication and observation requirements of the agents can be explicitly explored and connected to the system performance. Second, having agents adopt particular roles within a team can potentially provide further improvements in the learning speed. Finally, modifying the way in which agents estimate their ANT rewards can lead to substantial computational gains in addition to the already achieved speed up in the number of iterations required for convergence. We are currently investigating all three extensions of this work.

Acknowledgments

The authors would like to thank Matt Knudson for his insightful comments as well as his help with the preparation of this paper. This work was partially supported by AFOSR grant number FA9550-08-1-0187.

References

[1] Agogino, A. K. and Tumer, K., Handling communication restrictions and team formation in congestion games, *J. Auton. Agents Multi Agent Syst.* **13** (2006) 97–115.

[2] Agogino, A. K. and Tumer, K., Analyzing and visualizing multiagent rewards in dynamic and stochastic environments, *J. Auton. Agents Multi Agent Syst.* **17** (2008) 320–338.
[3] Agogino, A. K. and Tumer, K., Efficient evaluation functions for evolving coordination, *Evol. Comput.* **16** (2008) 257–288.
[4] Arai, S., Sycara, K. and Payne, T., Multi-agent reinforcement learning for planning and scheduling multiple goals, in *Proc. Fourth Int. Conf. on Multiagent Syst.* (2000), pp. 359–360.
[5] Arthur, W. B., Complexity in economic theory: Inductive reasoning and bounded rationality, *Am. Econ. Rev.* **84** (1994) 406–411.
[6] Chalkiadakis, G. and Boutilier, C., Coordination in multiagent reinforcement learning: A bayesian approach, in *Proc. Second Int. Joint Conf. on Autonomous Agents and Multiagent Systems (AAMAS-03)* (Melbourne, Australia, 2003).
[7] Challet, D. and Zhang, Y. C., On the minority game: Analytical and numerical studies, *Physica A* **256** (1998) 514.
[8] Claus, C. and Boutilier, C., The dynamics of reinforcement learning cooperative multiagent systems, in *Proc. Fifteenth National Conf. on Artificial Intelligence* (Madison, WI, 1998), pp. 746–752.
[9] de Cara, M. A. R., Pla, O. and Guinea, F., Competition, efficiency and collective behavior in the "El Farol" bar model, *Eur. Phys. J. B* **10** (1999) 187.
[10] Dietterich, T. G., Hierarchical reinforcement learning with the MAXQ value function decomposition, *J. Artif. Intell.* **13** (2000) 227–303.
[11] Guestrin, C., Lagoudakis, M. and Parr, R., Coordinated reinforcement learning, in *Proc. 19th Int. Conf. on Machine Learning* (2002).
[12] Hu, J. and Wellman, M. P., Multiagent reinforcement learning: Theoretical framework and an algorithm, in *Proc. Fifteenth Int. Conf. on Machine Learning* (1998), pp. 242–250.
[13] Hu, J. and Wellman, M. P., Online learning about other agents in a dynamic multiagent system, in *Proc. Second Int. Conf. on Autonomous Agents* (1998), pp. 239–246.
[14] Jefferies, P., Hart, M. L. and Johnson, N. F., Deterministic dynamics in the minority game, *Phys. Rev. E* **65**(016105) (2002).
[15] Jennings, N. R., Sycara, K. and Wooldridge, M., A roadmap of agent research and development, *Auton. Agents Multi-Agent Syst.* **1** (1998) 7–38.
[16] Kaelbling, L. P., Littman, M. L. and Moore, A. W., Reinforcement learning: A survey, *J. Artif. Intell. Res.* **4** (1996) 237–285.
[17] Khani, N. and Tumer, K., Fast multiagent learning: Cashing in on team knowledge, in *Artificial Neural Networks in Engineering* (ASME, St. Louis, 2008), pp. 3–10.
[18] Klügl, F., Bazzan, A. and Ossowski, S. (eds.), *Applications of Agent Technology in Traffic and Transportation* (Springer, 2005).
[19] Mataric, M. J., Coordination and learning in multi-robot systems, in *IEEE Intelligent Systems* (1998), pp. 6–8.
[20] McGlohon, M. and Sen, S., Learning to cooperate in multi-agent systems by combining Q-learning and evolutionary strategy, *Int. J. Lateral Comput.* **1** (2005) 58–64.
[21] Narendra, K. S. and Thathachar, M. A. L., *Learning Automata: An Introduction* (Prentice Hall, 1989).
[22] Panait, L., Tuyls, K. and Luke, S., Theoretical advantages of lenient learners: An evolutionary game theoretic perspective, *J. Mach. Learn. Res.* **9** (2008) 423–457.
[23] Parkes, D., On learnable mechanism design, in *Collectives and the Design of Complex Systems* (Springer, 2004).

[24] Pynadath, D. and Tambe, M., The communicative multiagent team decision problem: Analyzing teamwork theories and models, *J. Artif. Intell. Res.* **16** (2002) 389–423.
[25] Sherstov, A. and Stone, P., Three automated stock-trading agents: A comparative study, in *Agent Mediated Electronic Commerce VI: Theories for and Engineering of Distributed Mechanisms and Systems (AMEC 2004)*, Lecture Notes in Artificial Intelligence (Springer Verlag, Berlin, 2005), pp. 173–187.
[26] Stone, P., *Layered Learning in Multi-Agent Systems: A Winning Approach to Robotic Soccer* (MIT Press, Cambridge, MA, 2000).
[27] Stone, P., Sutton, R. S. and Kuhlmann, G., Reinforcement learning for RoboCup-soccer keepaway, *Adapt. Behav.* (2005).
[28] Sutton, R. S. and Barto, A. G., *Reinforcement Learning: An Introduction* (MIT Press, Cambridge, MA, 1998).
[29] Tambe, M., Towards flexible teamwork, *J. Artif. Intell. Res.* **7** (1997) 83–124.
[30] Taylor, M. E., Whiteson, S. and Stone, P., Comparing evolutionary and temporal difference methods for reinforcement learning, in *Proc. Genetic and Evolutionary Computation Conf.* (Seattle, WA, 2006), pp. 1321–1328.
[31] Tesauro, G., Practical issues in temporal difference learning, in *Advances in Neural Information Processing Systems*, Vol. 4, eds. Moody, J., Hanson, S. and Lippmann, R. (Morgan Kaufmann, 1992), pp. 259–266.
[32] Tumer, K. and Agogino, A., Distributed agent-based air traffic flow management, in *Proc. Sixth Int. Joint Conf. on Autonomous Agents and Multi-Agent Systems* (Honolulu, HI, 2007), pp. 330–337.
[33] Tumer, K., Agogino, A. and Wolpert, D., Learning sequences of actions in collectives of autonomous agents, in *Proc. First Int. Joint Conf. on Autonomous Agents and Multi-Agent Systems* (Bologna, Italy, 2002), pp. 378–385.
[34] Tumer, K., Welch, Z. T. and Agogino, A., Aligning social welfare and agent preferences to alleviate traffic congestion, in *Proc. Seventh Int. Joint Conf. on Autonomous Agents and Multi-Agent Systems* (Estoril, Portugal, 2008).
[35] Tumer, K. and Wolpert, D. (eds.), *Collectives and the Design of Complex Systems* (Springer, New York, 2004).
[36] Tumer, K. and Wolpert, D. H., Collective intelligence and Braess' paradox, in *Proc. Seventeenth National Conf. on Artificial Intelligence* (Austin, TX, 2000), pp. 104–109.
[37] Tuyls, K. and Parsons, S., What evolutionary game theory tells us about multiagent learning, *Artif. Intell.* **171** (2007) 406–416.
[38] Verbeeck, K., Nowe, A. and Tuyls, K., Coordinated exploration in multi-agent reinforcement learning: An application to load balancing, in *Proc. Fourth Int. Joint Conf. on Autonomous Agents and Multi-Agent Systems* (Utrecht, The Netherlands, 2005).
[39] Verbeeck, K., Peeters, M., Nowe, A. and Tuyls, K., Reinforcement learning in stochastic single and multi-stage games, in *Adaptive Agents and Multi-Agent Systems II*, Lecture Notes in Artificial Intelligence (Springer Verlag, Berlin, 2005), pp. 275–294.
[40] Vidal, J. M., Multiagent coordination using a distributed combinatorial auction, in *AAAI Workshop on Auction Mechanism for Robot Coordination* (2006).
[41] Vidal, J. M. and Durfee, E. H., The moving target function problem in multi-agent learning, in *Proc. Third Int. Conf. on Multi-Agent Systems* (AAAI/MIT press, 1998), pp. 317–324.
[42] Watkins, C. and Dayan, P., Q-learning, *Mach. Learn.* **8** (1992) 279–292.

[43] Wellman, M. P., Cheng, S.-F., Reeves, D. M. and Lochne, K. M., Trading agents competing: Performance, progress, and market effectiveness, *IEEE Intell. Syst.* **18** (2003) 48–53.
[44] Whiteson, S., Taylor, M. E. and Stone, P., Empirical studies in action selection for reinforcement learning, *Adapt. Behav.* **15** (2007).
[45] Wolpert, D. H. and Tumer, K., Optimal reward functions for members of collectives, *Adv. Complex Syst.* **4** (2001) 265–279.

Advances in Complex Systems, Vol. 12, Nos. 4 & 5 (2009) 475–492

World Scientific
www.worldscientific.com

MULTIAGENT LEARNING FOR BLACK BOX SYSTEM REWARD FUNCTIONS

KAGAN TUMER

Oregon State University, 204 Rogers Hall,
Corvallis, Oregon 97331, USA
kagan.tumer@oregonstate.edu

ADRIAN AGOGINO

UCSC, NASA Ames Research Center, Mailstop 269-3,
Moffett Field, California 94035, USA
adrian@email.arc.nasa.gov

Received 31 January 2009
Revised 22 May 2009

In large, distributed systems composed of adaptive and interactive components (agents), ensuring the coordination among the agents so that the system achieves certain performance objectives is a challenging proposition. The key difficulty to overcome in such systems is one of credit assignment: How to apportion credit (or blame) to a particular agent based on the performance of the entire system. In this paper, we show how this problem can be solved in general for a large class of reward functions whose analytical form may be unknown (hence "black box" reward). This method combines the salient features of global solutions (e.g. "team games") which are broadly applicable but provide poor solutions in large problems with those of local solutions (e.g. "difference rewards") which learn quickly, but can be computationally burdensome. We introduce two estimates for local rewards for a class of problems where the mapping from the agent actions to system reward functions can be decomposed into a linear combination of nonlinear functions of the agents' actions. We test our method's performance on a distributed marketing problem and an air traffic flow management problem and show a 44% performance improvement over team games and a speedup of order n for difference rewards (for an n agent system).

Keywords: Multiagent learning; black box reward functions; multiagent coordination.

1. Introduction

The ability of a team of agents to learn distributed policies has been demonstrated successfully in numerous domains such as controlling multiple robots, aggregating information from distributed data sources, and distributed system administration [11, 14, 17, 27, 30]. While diverse, each of these domains share two important properties fundamental to interesting distributed learning problems: (1) each agent learns its own set of actions (policy), (2) each policy is trying to maximize a system reward that is a nonlinear function of all the policies, thus coupling the policies

together. This type of problem is best described as a multiagent learning problem, where each agent, i, takes an action z_i and tries to maximize a reward function, $G(z)$, that is a function of z, the actions of all the agents [35, 33, 39, 36].

When the agent actions need to be coordinated, this issue becomes particularly challenging due to the structural credit assignment problem [1, 22, 37, 38]. In this problem, credit must be assigned to a particular agent based on the performance of the full system. For example, when an agent takes an action and G improves, the agent needs to determine whether its action was (partly) responsible for that improvement. Though lengthy learning trails can statistically eliminate the impact of other agents on G, such an approach is not practical for large systems. If G is linearly separable in the agents' actions, this credit assignment problem is trivial as each agent can maximize its own separate component of that reward. In contrast, if G depends on all the agents' actions directly, such as the parity problem, finding an adequate distributed solution is nearly impossible, and the problem needs to be reformulated. In this paper, we focus on problems where moderate numbers of agents need to coordinate their actions with one another to reach satisfactory values of G.

For systems with few agents, this credit assignment problem can be sidestepped, and all agents can use G directly. However, when the number of agents in a system increase, this method breaks down and agents need to receive a reward that accounts for their contribution to the system. The "difference reward" provides such a reward, and has produced good results in many domains [5, 31, 33, 34]. However, as currently expressed, the difference reward requires knowledge of the functional form of the system reward.

In this paper, we present an approach that lifts this requirement using two estimates of the difference reward that retains its fast learning characteristics, but does not require full knowledge of the functional form of $G(z)$. In Sec. 2, we briefly describe the related work. Section 3 describes the system reward structure and the basic difference used in multiagent learning. Section 4 derives two estimates for the difference reward that allows its application to domains with G of unknown functional form. Section 5 presents experimental results in both distributed marketing problem, and a complex air traffic flow problem. Section 6 discusses the mathematical implications and the future applications of the estimated difference rewards.

2. Related Work

In general, work in multiagent learning can be grouped into one of two broad categories: (i) work leveraging domain knowledge; and (ii) general work applicable to a subset of the domains. Some of the most successful work in multiagent learning fall into the first category. In robotic soccer for example, player specific subtasks, followed by tiling provide good convergence properties [27]. In foraging robot coordination, specific rules induce good division of labor [19]. In a distributed air traffic

control domain, a combination of positive rewards and penalty rewards allows a collection of aircrafts to navigate safely [15]. In all cases, the agent coordination is achieved through exploiting knowledge of the system dynamics and accentuating the known desirable interactions among the agents.

The second set of approaches provide general solutions to a subset of the problems. Early work on this topic focused on "team games" where each agent considers itself the only agent in the system and receives the full system reward. An example of this approach is the control of four elevators where a separate reinforcement learner was used to control each elevator, and each learner received the full system reward [10]. While such a "team game" approach is effective, it is restricted to domains with a small number of agents. In problems where groups of agents can be assumed to be independent, the task can be decomposed by learning a set of basis functions used to represent the value function, where each basis only processes a small number of the state variables [14]. Task decomposition has also been used in single agent RL using hierarchical reinforcement learning methods such as MAXQ value function decomposition [12]. In multiagent learning, Partially Observable Markov Decision Processes (POMDPs) can be simplified through piecewise linear rewards [24]. In other cases, agents can be assumed to be locally connected through a graph and can learn efficiently through local rewards [6]. Outside of reinforcement learning, mechanism design has been used with MDPs to address the issue of creating good agent incentives for specific types of rewards [25].

3. Agent and System Rewards

As stated in the introduction, in this paper we present a method to estimate difference rewards that does not require full knowledge of the functional form of the system reward G.

3.1. *System reward*

In particular, we focus our study to the class of problems where system reward is in the form:

$$G(z) = G_f(f(z)) = G_f\left(\sum_i f_i(z_i)\right), \tag{1}$$

where G_f is a known nonlinear function and the f_is are unknown nonlinear functions. Table 1 summarizes the functional form of G and its arguments.

Table 1. Functional forms for system objective function.

Function	Form		Argument
G	Unknown	Nonlinear	z
G_f	Known	Nonlinear	f
f	Known	Linear	f_i
f_i	Unknown	Nonlinear	z_i

The key assumption in this work is that the f_i cannot be sampled from the domain, but that $\sum_i f_i$ can be sampled (potentially at a high cost). This form of $G(z)$ applies to a large number of domains where agents have an unknown effect on their environment (f_i) and these effects are aggregated together. Such domains include air (or highway) traffic flow management, distributed gating, and distributed information gathering. While the agents do not know the f_is they do know how these aggregated effects contribute to the system goal in the form of G_f. Our estimate exploits this structure of G to create local rewards that allow learning to proceed significantly faster than directly using G and be applied to systems where the agent-specific rewards cannot be applied because the form of G is unknown.

3.2. *Difference reward*

In a multiagent setting, while each agent can try to maximize the system reward directly, such an approach leads to slow/poor learning due to the structural credit assignment problem. An alternative is to have each agent attempt to maximize an agent-specific reward function derived in such a way that if agents succeed in maximizing that reward function, they collectively also maximize G. One such reward function is the **difference** reward function of the form [33]:

$$D_i \equiv G(z) - G(z - z_i + c_i), \tag{2}$$

where z_i is the action of agent i, and c_i is an arbitrary "action" that does not depend on agent i's actions.[a] In the second term of D_i, $z - z_i + c_i$ represents the "counterfactual" states where the action of agent i, z_i, is replaced by a fixed action c_i that is independent of the agent's action.

There are two advantages in using D: First, the second term, $G(z - z_i + c_i)$, differs from the first term, $G(z)$, only in the actions of agent i. If agent i's action is not tightly coupled to the actions of the other agents, then the second term will subtract out much of the impact of the actions of the other agents in the system, therefore providing an agent with a "cleaner" signal than G. For instance if all the other agents choose poor actions, the impact of these actions would appear in both terms of D_i, and would mostly cancel out. This benefit has been dubbed "learnability" (agents have an easier time learning) in previous work [33]. Second, because the second term does not depend on the actions of agent i, any action taken by agent i that improves D, also improves G. Therefore, we expect policies that maximize D will also maximize G. This specific form of difference reward has been effective in a number of domains including congestion problems, multi-rover policy evolution, and bin-packing [2, 30, 33].

[a]This notation uses zero padding and vector addition rather than concatenation to form full state vectors from partial state vectors.

As an example, consider the application of this reward to a multi-robot coordination problem where multiple robots need to gather importance weighted information and maximize the total information collected by all the robots [5, 30]. In such a case, selecting a c_i that removes the robots' observations from the system, the difference reward measures the contribution of that robot to the system. Note, this is not equivalent to having each robot simply maximize the information it collects (which leads to poor system behavior) [5]. Instead, the difference reward leads to robots exploring areas that would not have been explored by other robots. That is, if a second robot would have observed a particular area, then the difference reward provides low values, urging the robot to find information with more value to the full system [5].

4. Estimates of Difference Rewards

Though providing a good compromise between aiming for system performance and removing the impact of other agents from an agent's reward, one issue that may plague D is computational cost. Because it relies on the computation of the counterfactual term $G(z - z_i + c_i)$ (i.e. the system performance without agent i) it may be difficult or impossible to compute, particularly when the exact mathematical form of G is not known.

For reward functions that are of the form given in Eq. (1) and summarized in Table 1, however, we can derive estimates for D that overcome this limitation. Our premise is that we can sample values from $f(z)$, enabling us to compute G, but that we cannot sample from each $f_i(z_i)$. In addition, we assume we may not be able to even compute $f(z)$ directly and must sample it from a "black box" computation (e.g. a system simulator) or measure it from the environment.

4.1. *First estimate*

The key element in the computation of the difference reward is the counterfactual $G(z - z_i + c_i)$:

$$\begin{aligned} G(z - z_i + c_i) &= G_f(f(z - z_i + c_i)) \\ &= G_f\left(\sum_{j \neq i} f_j(z_j) + f_i(c_i)\right) \\ &= G_f(f(z) - f_i(z_i) + f_i(c_i)). \end{aligned} \tag{3}$$

Unfortunately, we cannot compute this directly as the values of $f_i(z_i)$ are unknown. However, if agents take actions independently (i.e. they do not observe how other agents act before taking their own actions) we can take advantage of the linear form of $f(z)$ in the f_is with the following equality:

$$E(f_{-i}(z_{-i}) \,|\, z_i) = E(f_{-i}(z_{-i}) \,|\, c_i), \tag{4}$$

where $E(f_{-i}(z_{-i}) \mid z_i)$ is the expected value of $f_{j\neq i}$ (all fs other than f_i) given the value of z_i and $E(f_{-i}(z_{-i}) \mid c_i)$ is the expected value of $f_{j\neq i}$ given c_i. We then get the following estimate for $f(z - z_i + c_i)$:

$$\begin{aligned} f(z - z_i + c_i) &= f(z) - f_i(z_i) + f_i(c_i) \\ &= f(z) - f_i(z_i) - E(f_{-i}(z_{-i})|z_i) \\ &\quad + f_i(c_i) + E(f_{-i}(z_{-i})|c_i) \\ &= f(z) - E(f_i(z_i)|z_i) - E(f_{-i}(z_{-i})|z_i) \\ &\quad + E(f_i(c_i)|c_i) + E(f_{-i}(z_{-i})|c_i) \\ &= f(z) - E(f(z)|z_i) + E(f(z)|c_i). \end{aligned} \tag{5}$$

Therefore, we can evaluate $D_i = G(z) - G(z - z_i + c_i)$ as:

$$D_i^{\text{est1}} = G_f(f(z)) - G_f(f(z) - E(f(z)|z_i) + E(f(z)|c_i)).$$

The first term of D_i^{est1} is the same as the original difference reward. The second term of D_i^{est1} tries to remove the impact of the other agents, but cannot do this as elegantly as the difference reward since the form of function $f(z)$ is not known. Instead of subtracting out $f_i(z_i)$ and adding $f_i(c_i)$ directly, we estimate this by taking the difference between average impact of action z_i of $f(z)$ and the average impact of action c_i on $f(x)$. This leaves us with the task of estimating the values of $E(f(z)|z_i)$ and $E(f(z)|c_i)$). These estimates can be computed by keeping a table of averages where we average the values of the observed $f(z)$ for each value of z_i that we have seen. Note, this estimate improves as the number of samples increases.

4.2. *Second estimate*

The discussion above is generally applicable to any selection of c_i. We can improve this estimate if we set $c_i = E(z_i)$ and make the mean squared approximation of $f_i(E(z)) \approx E(f_i(z))$. The last expectation in D_i^{est1} is transformed as follows:

$$\begin{aligned} E(f(z)|c_i) &= E\left(\left(f_i(z_i) + \sum_{j\neq i} f_j(z_j)\right)\middle| E(z_i)\right) \\ &= E(f_i(z_i)|E(z_i)) + \sum_{j\neq i} E(f_j(z_j)) \\ &= E(f_i(E(z_i))) + \sum_{j\neq i} E(f_j(z_j)) \\ &\approx E(E(f_i(z_i))) + \sum_{j\neq i} E(f_j(z_j)) \\ &= \sum_j E(f_j(z_j)) \\ &= E(f(z)). \end{aligned} \tag{6}$$

We then we can estimate $G(z) - G(z - z_i + c_i)$ as:

$$D_i^{est2} = G_f(f(z)) - G_f(f(z) - E(f(z)|z_i) + E(f(z))).$$

The estimate D_i^{est2} is the same as D_i^{est1}, except that $E(f(z)|c_i)$ has been replaced with $E(f(z))$. This formulation has two advantages over D_i^{est1}: First, there are more samples at our disposal to estimate $E(f(z))$ than we do to estimate $E(f(z)|c_i))$. Second, this removes the need to select a value for c_i. Since selecting a value of c_i that will lead to high performance can be difficult in some domains, it can be advantageous to have this parameter removed.

5. Experimental Results

To test the effectiveness of the difference reward and its estimates, we conduct a series of experiments in two domains. The first domain is an illustrative example in the form of a distributed marketing problem, where separate marketing agents try to market a common resource to distinct groups of potential customers. The second domain tests the performance of our reward system in a complex air traffic flow domain, where we use the Future ATM Concepts Evaluation Tool (FACET) air traffic simulator to test the ability of learning agents to create policies that reduce congestion while minimizing delays [8]. In all experiments, we test the performance of five different methods. The first method is Monte Carlo (MC) estimation, where random policies are created, with the best policy being chosen. The other four methods are based on reinforcement learning agents where the agents are maximizing one of the following rewards:

(1) the system reward, $G(z)$;
(2) the actual difference reward, $D_i(z)$;
(3) the first difference reward estimate, $D_i^{\text{est1}}(z)$; and
(4) the second difference reward estimate, $D_i^{\text{est2}}(z)$.

In these experiments, the aim of each agent is to learn to take actions that will lead to the best system performance, G. To form policies, each agent uses an agent-specific reward function and tries to maximize it with its own reinforcement learner [20, 40] (though alternatives such as evolving neuro-controllers are also effective [28, 30]). To clearly illustrate the benefit of the reward estimates, in this paper we focus on domains that only need to utilize immediate rewards. As a consequence, simple table-based immediate reward reinforcement learning is used. The reinforcement learner is equivalent to an ϵ-greedy Q-learner with a discount rate of 0 [20]. In all the experiments, the learning rate is equal to 0.5 and ϵ is equal to 0.25. Note that in many domains, reinforcement learning needs to look at rewards beyond the immediate reward and address a temporal credit assignment problem of how to reward a current action for a sequence of future rewards. Difference rewards

have been shown to address both the structural and temporal credit assignment problems for domains where the functional form of G is known [4].

To make the agent results comparable to the MC estimation, the best policies chosen by the agents over a single trial are used in the results. MC and similar random approaches are common in complex air traffic problems [21]. All results are an average of 30 independent trials with the differences in the mean ($\sigma/\sqrt{n}$) shown as error bars, though in most cases the error bars are too small to see.

5.1. *Distributed marketing problem*

The first domain we study is a distributed marketing problem where a number of agents need to choose a strategy, and their reward depends on the strategies of all the agents. This is a form of congestion game where particular joint actions lead to desirable or undesirable behavior based on the number of other agents that have selected that particular action [4, 9, 16, 18, 32, 41].

5.1.1. *Problem description*

In the "Marketing Problem" there are n agents marketing a constrained resource to n different demographics. Examples include marketing a resort hotel to several different parts of the country, or a public transportation system to different cities in a metropolitan area. In this problem, each agent has a finite set of marketing strategies. The action of agent, i, is to choose a strategy z_i. The number of people who use the resource is an unknown nonlinear function of the marketing of all the agents, $f(z)$. After each training episode, the value of $f(z)$ is measured. We assume that the marketers are targeting disjoint groups so $f(z)$ has the following form: $f(z) = \sum_i f_i(z_i)$, where $f_i(z_i)$ is a nonlinear function of agent i's marketing action. The function $f(z)$ represents the aggregate sum of the effects of all the agents marketing actions.

This form represents situations where marketers do not have a model for the effects of their marketing, so the function $f_i(z_i)$ is unknown. In addition, the value of $f_i(z_i)$ is never measured as we only have measurements of the aggregate number of people using the resource, $f(z)$. The system goal is to have the optimal amount of people use the resource. More revenue is gained by having more people use it, but we do not want it to become overused as that will hurt our reputation (e.g. overuse a transportation domain would result in congestion). The system goal is represented by a known nonlinear function of the total number of people using the resource: $G(z) = G_f(f(z))$. Note that while the functional form of G_f is known, the form of G (in terms of z) is not, since $f(z)$ is unknown.

5.1.2. *Results*

We conducted a series of experiments where agents choose one of M marketing actions, where an action deterministically resulted in zero to C customers using the resource, depending on the action. For each agent, the mapping from action to

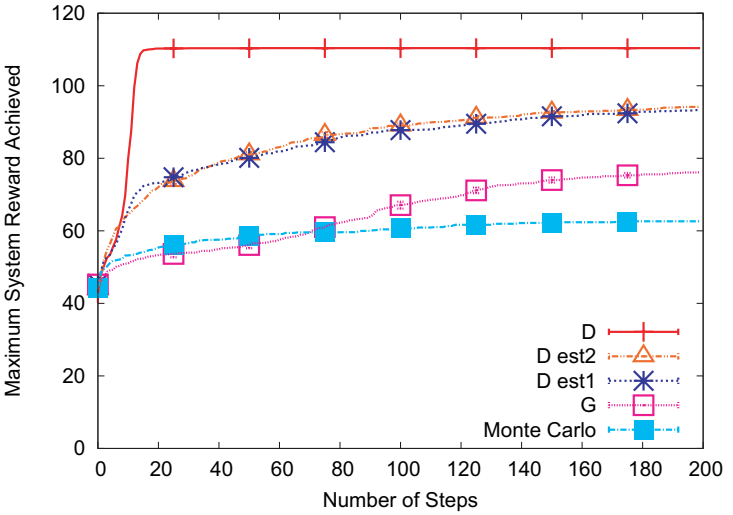

Fig. 1. Marketing experiment with 100 agents. The estimates for D perform better than G, though not as well as the full D, which is not "computable" in many real world domains.

customer response $f_i(z)$ was chosen at random at the start of the experiment. The function G_f was set to $ke^{-k/c}$ where $k = f(z)$ is the aggregate number of customers that use the resource and c is the optimal capacity for the resource.

Figure 1 shows the performance of the five different methods in a marketing problem with 100 agents, where $M = 10$ and $C = 18$. MC optimization provides a baseline solution. Agents using G directly as their reward perform slightly better than MC. But in this case, each agent's reward is affected by the actions of the other 99 other agents, making it hard for an agent to discern the effects of its action on its reward. In contrast agents using the true difference reward learn fast and learn well. However, this reward is not directly computable when agents do not know the functional form of $f(z)$ needed to compute the counterfactual $G_f(f(z - z_i + c_i))$. Therefore, this is a theoretical result that cannot be implemented in many real domains. The results for the estimates to the difference rewards (described in Sec. 4), where an agent only needs to know the value of $f(z)$, are promising as they outperform agents using G.

One interesting question that arises concerns the convergence properties of agents using the different rewards. Unfortunately analysis of convergence in a multiagent problem is difficult given nonlinear interactions between the agent learning algorithms and the reward function. In theory, the difference rewards are shown to converge to (potentially local) minima as long as the system reward converges [33]. In practice, the performance of the agents to not change significantly after 200 learning steps, in these experiments.

Figure 2 shows the scaling results for the number of agents ranging from 1 to 200. These results show that the relative performance of the algorithms is not affected by the number of agents. Note that the true difference reward has remarkably good

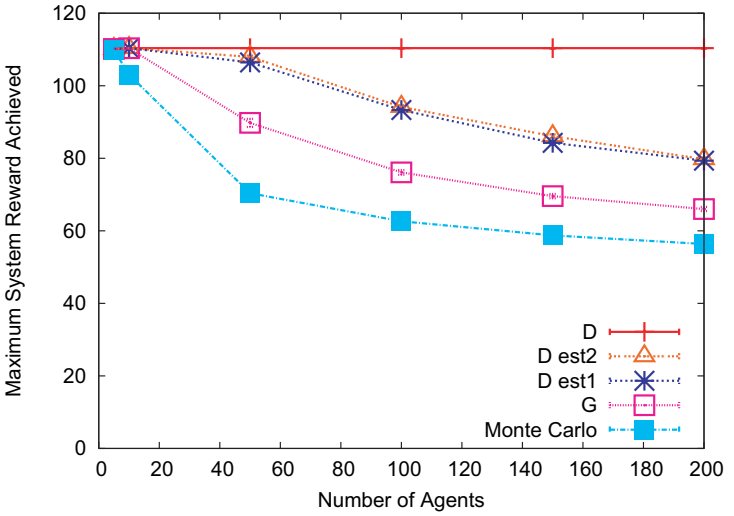

Fig. 2. Marketing experiment scaling after 200 episodes. The estimates for D degrade more gracefully than G.

scaling characteristics as its performance does not degrade as the number of agents is increased from five agents to 200 agents, making it a good choice for large domains where the functional form of G is known.

5.1.3. *Computational cost of D and D^{est}*

The results above show the performance of the different algorithms after a specific number of episodes, demonstrating that D performs significantly better than the other algorithms. In domains where the functional form of G is known, D can often be computed without explicit calls to G [2]. However, if the agents are unable to streamline their computation of D, agents using the difference reward may be forced to make many computations of G. In general, for n agents, that means D gets n times as many G function calls. Table 2 shows the relative performance for a given number of G evaluations. The reward D performs best when used over the full 200 episodes, but requires 4000 computations of G. The two estimates to D provide

Table 2. Marketing experiment with 100 agents, after 200 G evaluations (except for D^{4000} which has 4000 G evaluations at episode 200).

Reward	G	$\sigma/\sqrt{n}$	Steps
D^{est2}	94.2	0.5	200
D^{est1}	93.3	0.5	200
D	52.6	0.8	2
D^{4000}	110.4	0.0003	200
G	76.1	0.7	200
MC	62.6	0.6	200

the best compromise between performance and computational cost, outperforming both D and G for a given number of G evaluations. Note that in cases where G is only computed by sampling the environment, it may not even be possible to compute D at any computational cost and the estimates will have to be used as discussed in Sec. 6.

5.2. *Air traffic flow problem*

The second domain we study is the complex domain of air traffic flow management [3, 7, 13, 23, 26, 29, 31]. This is a complex real world problem, where the agent actions cannot be directly be mapped to a system reward in analytical form, creating a "black box" reward for the learning system.

5.2.1. *Problem description*

In this section, we summarize how distributed learning agents can learn to manage air traffic flow [31]. First, we will assign agents to airspace locations called "fixes" to map the air traffic problem to a multiagent problem. Each agent is responsible for any aircraft going through its fix [3, 31]. The action of an agent is to determine the separation (distance between aircraft) that aircraft have to maintain, when going through the agent's fix (though aircraft will always keep a safe distance, d_s, if d is set too low). The effect of issuing higher separation values is to slow down the rate of aircraft that go through the fix. By increasing the value of d, an agent can limit the amount of air traffic downstream of its fix, reducing congestion at the expense of increasing the delays upstream.

Second, we will use FACET (Future ATM Concepts Evaluation Tool, where ATM stands for Air Traffic Management) to simulate air traffic and determine the impact of the agents' actions [8]. FACET simulates air traffic based on flight plans and through a graphical user interface allows the user to analyze congestion patterns of different sectors and centers (Fig. 3). FACET also allows the user to change the flow patterns of the aircraft through a number of mechanisms, including "metering" aircraft. Metering is performed by choosing a "Miles in Trail" (MIT) value, which specifies the minimum distance that aircraft may be spaced from each other when passing through a particular location. Larger MIT values cause aircraft to be spaced further apart. In this paper, agents send scripts to FACET asking it to simulate air traffic based on metering orders imposed by the agents. The agents then produce their rewards based on received feedback from FACET about the impact of these meterings.

Finally, we will define a system reward function that focuses on the amount of congestion in a particular sector and on the amount of measured air traffic delay. This is measured as a function of the agents' action vector z, specifying the MIT values chosen by the agents. More precisely, we have:

$$G(z) = -((1-\alpha)B(z) + \alpha C(z)), \tag{7}$$

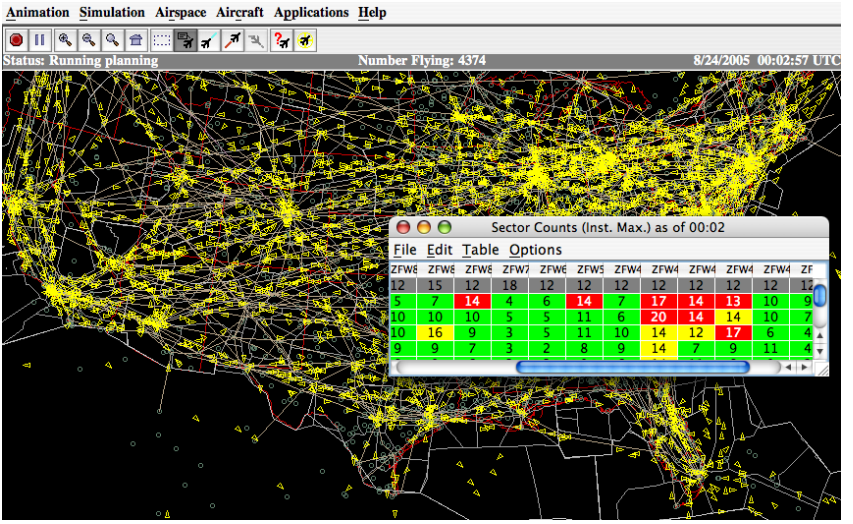

Fig. 3. FACET screen-shot displaying traffic routes.

where $B(z)$ is the total delay penalty for all aircraft in the system, and $C(z)$ is the total congestion penalty, and α determines the relative importance of these two. Neither $B(z)$, nor $C(z)$ can be analytically computed by an agent. Rather, they are computed after the number of aircraft in a sector are computed.

With $\alpha = 0.5$, for the two-congestion problem in our experiments we used an instance of this reward function described in detail in Ref. 31 and summarized as follows:

$$G(z) = -A_1 \sum_i \sum_t u(t - T_i) k_{t,i} (t - T_i) - A_2 \sum_i \sum_t u(k_{t,1} - C_i) e^{\beta(k_{t,1} - C_i)}, \qquad (8)$$

where t is time, $k_{t,i}$ is the number of aircraft in congestion i, and $u(t)$ is the unit step function. T_i is the delay penalty constant ($T_1 = 200$ and $T_2 = 175$ here) and C_i is the congestion penalty constant ($C_1 = 18$ and $C_2 = 15$ here). A_1 and A_2 are scaling factors for the delay and congestion terms ($A_1 = \frac{1}{2}$ and $A_2 = 50$), and $\beta = 0.3$. The values of $k_{t,i}$ are computed by FACET and are affected by the actions of the agents as described in the following section. The term β is a user defined constant controlling the penalty curve for congestion. Note that G cannot be expressed in closed form in terms of the actions of the agents, since the effect of those actions of the congestion ($k_{t,j}$) is not known in closed form.

5.2.2. *Results*

We tested the performance of the different rewards on an air traffic domain with 300 aircraft. The aircraft go through two points of congestion over a four hour simulation, with 200 going over one point of congestion and 100 going over the other point of congestion. The second congestion is less severe than the first one,

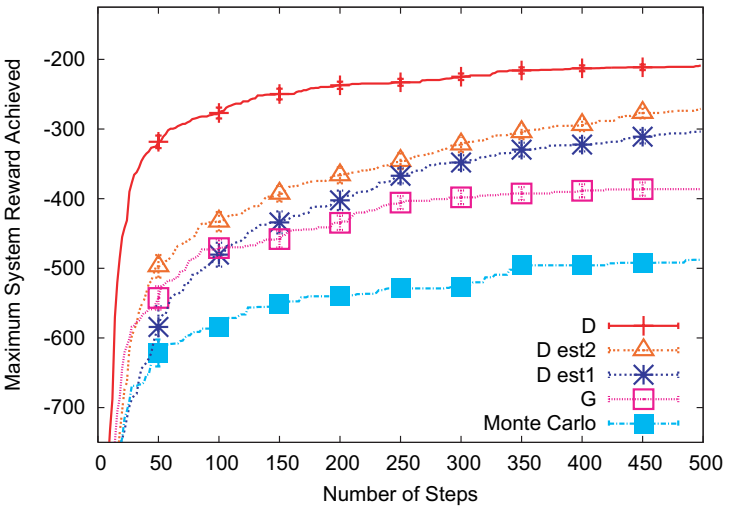

Fig. 4. Performance with 300 aircraft, 20 agents. The estimates for D perform better than G, though not as well as the full D, which is computationally expensive in this domain.

so agents have to form different policies depending which point of congestion they are influencing. The points of congestion are created by setting up a series of flight plans that cause the number of aircraft in the sectors of interest to be significantly more than the number allowed by the FAA.

The results displayed in Fig. 4 show that the relative performance of the five methods is similar to the Marketing Problem. However, in this case D^{est2} performs better than D^{est1}. This is caused by the limited amount of data available in this domain and that D^{est2} draws from a larger sample to estimate D, resulting in a cleaner signal. Figure 5 shows scaling results for the number of agents varying from 10 to 50 and shows that the conclusions are not sensitive to the number of agents. Agents using D^{est2} perform slightly better than agents using D^{est1} in all cases but for 40 and 50 agents where they are statistically equivalent. While adding more fixes increases the amount of control the agents have over the system, this increase does not necessarily improve performance. The main issue is that when the number of fixes grows in this problem, the number of aircraft going through each fix decreases. This could result in certain fixes in superior positions to control less aircraft, causing a reduction in performance.

5.2.3. *Computational cost of D and D^{est}*

As was the case for the Marketing domain, the results above show that D is superior to the other algorithms. However, in the air traffic domain, D can only be computed with additional calls to the FACET simulator, which come at significant computational cost. The computation cost of the system reward, G [Eq. (7)] is almost entirely dependent on the computation of the airplane counts for the congestions

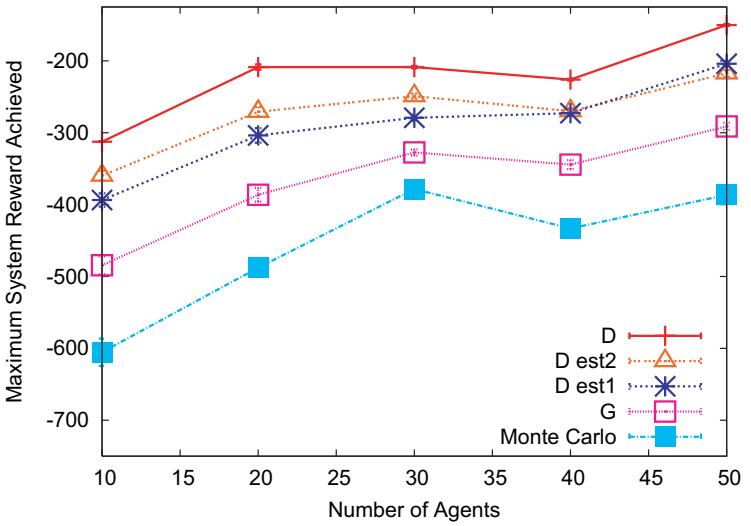

Fig. 5. Impact of number of agents on system performance with 300 aircraft. Performance improves with higher number of agents, but only if the algorithms and agents rewards can "extract" the extra information.

Table 3. System performance for 20 agents, 300 aircraft, after 2100 G evaluations (except for D^{44K} which has 44,100 G evaluations at step 2100).

Reward	G	$\sigma/\sqrt{n}$	Steps
D^{est2}	−232.5	7.55	2100
D^{est1}	−234.4	6.83	2100
D	−277.0	7.80	100
D^{44K}	−219.9	4.48	2100
G	−412.6	13.60	2100
MC	−639.0	16.40	2100

k_t, which need to be computed using FACET.[b] Except when D is used, the values of k are computed once per episode. However, to compute the counterfactual term in D, if FACET is treated as a "black box," each agent has to compute its own values of k for their counterfactual resulting in $n+1$ computations of k per episode.

Table 3 shows the performance of the algorithms after 2100 G computations for each of the algorithms for the simulations presented in Fig. 4 where there were 20 agents and two congestions. All the algorithms except the fully computed D reach 2100 k computations at time step 2100. D, however, computes k once for the system, and then once for each agent, leading to 21 computations per time step. It therefore reaches 2100 computations at time step 100. We also show the results of the full D computation at $t = 2100$, which needs 44,100 computations

[b]In our simulations a computation from FACET took 900 milliseconds, while all the other computation for all 20 agents in an episode took a combined 5 milliseconds.

of k as D^{44K}. Although D^{44K} provides the best result by a slight margin, it is achieved at a considerable computational cost. Indeed, the performance of the two D estimates is remarkable in this case as they were obtained with about 20 times fewer computations of k. Furthermore, the two D estimates, significantly outperform the full D computation for a given number of computations of k and validate the assumptions made in Sec. 4. This shows that for this domain, in practice it is more fruitful to perform more learning steps and approximate D, than few learning steps with full D computation when we treat FACET as a black box.

6. Discussion

Learning multiagent policies is difficult due to the structural credit assignment problem of how to credit an action's contribution to a system reward, which is a function of many actions. Furthermore, the mapping from agent actions to system reward cannot always be computed in closed form. This paper proposes to address this issue using an estimate to a "difference reward" where agents learn using an agent-centric reward that promotes coordination. On a marketing problem and an air traffic flow problem, experimental results show that our method provides an improvement in performance by up to 44% over team games and difference rewards (when computational cost is taken into account).

Whether the difference reward or its estimate should be used depends on what is known about the functional form of the system reward, and how much it costs to compute. We are interested in three main types of system reward:

(1) the system reward has a functional form that is completely known;
(2) the system reward is a black box with high computational costs; or
(3) the system reward is sampled from the environment, where we cannot demand samples for arbitrary actions.

The air traffic flow management problem is an instance of the second type of problem, since the FACET simulator can be used as a black box to retrieve values of $f(z)$, but at an extremely high computational cost. In this case, agents should use the estimate of the difference reward to save computational costs. The marketing problem is an instance of the third type because agents do not know f_i (i.e. they do not know how their actions affect their target audience). In this case, the agents can only count the aggregate number of people affected by all the agents, so they must use the estimate to the difference reward, since the true difference reward cannot be computed. (Note that the marketing problem would be the first type of problem if the values of f_i were known ahead of time and the difference reward could be computed in closed form. In this case, the true difference reward can be used.)

This work provided the groundwork for multiagent learning in domains where the system reward is not known in closed form. There are three promising extensions of this work: First, the manner in which the estimate for $f(z)$ used in the difference rewards is computed can be improved. Currently, we use simple averaging, though

using data aging or similarity measure to provide a weighted average can improve the estimate. Second, the functional form of the system rewards can be extended beyond that given in Eq. (1), and use more general machine learning methods to estimate the difference reward. Third, the difference reward estimates are now restricted by the form of G. Blending imperfect models of the environment with true samples in order to compute the difference reward would increase both the speed and the accuracy of the estimates. We are currently investigating all three avenues of research and extending the application domains to include robotic exploration and more realistic forms of the air traffic flow problem (including the role of human air traffic controllers).

Acknowledgments

The authors thank Dr. Banavar Sridhar and Shon Grabbe for their help with air traffic flow management and the FACET simulator. This work was partially supported by AFOSR grant number FA9550-08-1-0187.

References

[1] Agogino, A. and Tumer, K., Unifying temporal and structural credit assignment problems, in *Proc. Third Int. Joint Conf. on Autonomous Agents and Multi-Agent Systems* (New York, NY, 2004).

[2] Agogino, A. and Tumer, K., QUICR-learning for multiagent coordination, in *Proc. 21st National Conf. on Artificial Intelligence* (Boston, MA, 2006).

[3] Agogino, A. and Tumer, K., Regulating air traffic flow with coupled agents, in *Proc. Seventh Int. Joint Conf. on Autonomous Agents and Multi-Agent Systems* (Estoril, Portugal, 2008).

[4] Agogino, A. K. and Tumer, K., Handling communication restrictions and team formation in congestion games, *J. Auton. Agents Multi Agent Syst.* **13** (2006) 97–115.

[5] Agogino, A. K. and Tumer, K., Efficient evaluation functions for evolving coordination, *Evol. Comput.* **16** (2008) 257–288.

[6] Bagnell, J. A. and Ng, A. Y., On local rewards and the scalability of distributed reinforcement learning, *News Physiol. Sci.* **18** (2006).

[7] Bayen, A. M., Grieder, P., Meyer, G. and Tomlin, C. J., Lagrangian delay predictive model for sector-based air traffic flow, *AIAA J. Guid. Cont. Dyn.* **28** (2005) 1015–1026.

[8] Bilimoria, K. D., Sridhar, B., Chatterji, G. B., Shethand, K. S. and Grabbe, S. R., Facet: Future atm concepts evaluation tool, *Air Traffic Cont. Q.* **9** (2001).

[9] Challet, D. and Zhang, Y. C., On the minority game: Analytical and numerical studies, *Physica A* **256** (1998) 514.

[10] Crites, R. H. and Barto, A. G., Improving elevator performance using reinforcement learning, in *Advances in Neural Information Processing Systems*, eds. Touretzky, D. S., Mozer, M. C. and Hasselmo, M. E., Vol. 8 (MIT Press, 1996), pp. 1017–1023.

[11] de Oliveira, D., Ferreira, P. R. Jr., and Bazzan, A. L. C., A swarm based approach for task allocation in dynamic agents organizations, in *AAMAS '04: Proc. Third Int. Joint Conf. on Autonomous Agents and Multiagent Systems* (IEEE Computer Society, Washington, DC, USA, 2004), pp. 1252–1253.

[12] Dietterich, T. G., Hierarchical reinforcement learning with the maxq value function decomposition, *J. Artif. Intell.* **13** (2000) 227–303.
[13] Donohue, G. L. and Shaver III, R. D., *TERMINAL CHAOS: Why U.S. Air Travel Is Broken and How to Fix It* (Amer Inst of Aeronautics and Astronautics, 2008).
[14] Guestrin, C., Lagoudakis, M. and Parr, R., Coordinated reinforcement learning, in *Proc. 19th Int. Conf. on Machine Learning* (2002).
[15] Hill, J. C., Johnson, F. R., Archibald, J. K., Frost, R. L. and Stirling, W. C., A cooperative multiagent approach to free flight, in *AAMAS '05: Proc. Fourth Int. Joint Conf. on Autonomous Agents and Multiagent Systems* (ACM Press, New York, NY, USA, 2005), ISBN 1-59593-093-0, pp. 1083–1090.
[16] Jefferies, P., Hart, M. L. and Johnson, N. F., Deterministic dynamics in the minority game, *Phys. Rev. E* **65**(016105) (2002).
[17] Jennings, N. R., On agent-based software engineering, *Artif. Intell.* **177** (2000) 277–296.
[18] Johnson, N. F., Jarvis, S., Jonson, R., Cheung, P., Kwong, Y. R. and Hui, P. M., Volatility and agent adaptability in a self-organizing market (1998), preprint cond-mat/9802177.
[19] Jones, C. and Mataric, M. J., Adaptive division of labor in large-scale multi-robot systems, in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS-03)* (Las Vegas, NV, 2003), pp. 1969–1974.
[20] Kaelbling, L. P., Littman, M. L. and Moore, A. W., Reinforcement learning: A survey, *J. Artif. Intell. Res.* **4** (1996) 237–285.
[21] Lecchini, A., Glover, W., Lygeros, J. and Maciejowskia, J., Monte Carlo optimization strategies for air-traffic control, *AIAA Guidance, Navigation, and Control Conference* (San Francisco, CA, 2005), pp. 470–482.
[22] McGlohon, M. and Sen, S., Learning to cooperate in multiagent systems by combining Q-learning and evolutionary strategy, *Int. J. Lateral Comput.* **1** (2005) 58–64.
[23] Menon, P. K., Sweriduk, G. D. and Sridhar, B., Optimal strategies for free flight air traffic conflict resolution, *J. Guid. Cont. Dyn.* **22** (1999) 202–211.
[24] Nair, R., Tambe, M., Yokoo, M., Pynadath, D. and Marsella, S., Taming decentralized POMDPs: Towards efficient policy computation for multiagent settings, in *Proc. Eighteenth Int. Joint Conf. on Artificial Intelligence* (Acapulco, Mexico, 2003).
[25] Parkes, D. and Singh, S., An MDP-based approach to online mechanism design, *News Physiol. Sci.* **16** (2004) 791–798.
[26] Pechoucek, M., Sislak, D., Pavlicek, D. and Uller, M., Autonomous agents for air-traffic deconfliction, in *Proc. Fifth Int. Joint Conf. on Autonomous Agents and Multi-Agent Systems* (Hakodate, Japan, 2006).
[27] Stone, P., Sutton, R. S. and Kuhlmann, G., Reinforcement learning for RoboCup-soccer keepaway, *Adapt. Behav.* (2005).
[28] Taylor, M. E., Whiteson, S. and Stone, P., Comparing evolutionary and temporal difference methods for reinforcement learning, in *Proc. Genetic and Evolutionary Computation Conf.* (Seattle, WA, 2006), pp. 1321–1328.
[29] Tomlin, C., Pappas, G. and Sastry, S., Conflict resolution for air traffic management: A study in multiagent hybrid systems, *IEEE Trans. Automat. Cont.* **43** (1998) 509–521.
[30] Tumer, K. and Agogino, A., Coordinating multi-rover systems: Evaluation functions for dynamic and noisy environments, in *The Genetic and Evolutionary Computation Conference* (Washington, DC, 2005).
[31] Tumer, K. and Agogino, A., Distributed agent-based air traffic flow management, in *Proc. Sixth Int. Joint Conf. on Autonomous Agents and Multi-Agent Systems* (Honolulu, HI, 2007), pp. 330–337.

[32] Tumer, K., Welch, Z. T. and Agogino, A., Aligning social welfare and agent preferences to alleviate traffic congestion, in *Proc. Seventh Int. Joint Conf. on Autonomous Agents and Multi-Agent Systems* (Estoril, Portugal, 2008).
[33] Tumer, K. and Wolpert, D. (eds.), *Collectives and the Design of Complex Systems* (Springer, New York, 2004).
[34] Tumer, K. and Wolpert, D. H., Collective intelligence and Braess' paradox, in *Proc. Seventeenth National Conf. on Artificial Intelligence* (Austin, TX, 2000), pp. 104–109.
[35] Tuyls, K. and Parsons, S., What evolutionary game theory tells us about multiagent learning, *Artif. Intell.* **171** (2007).
[36] Verbeeck, K., Nowe, A. and Tuyls, K., Coordinated exploration in multiagent reinforcement learning: An application to loadbalancing, in *Proc. Fourth Int. Joint Conf. on Autonomous Agents and Multi-Agent Systems* (Utrecht, The Netherlands, 2005).
[37] Verbeeck, K., Peeters, M., Nowe, A. and Tuyls, K., Reinforcement learning in stochastic single and multi-stage games, in *Adaptive Agents and Multi-Agent Systems II*, Lecture Notes in Artificial Intelligence (Springer Verlag, Berlin, 2005), pp. 275–294.
[38] Vidal, J. M., Multiagent coordination using a distributed combinatorial auction, in *AAAI Workshop on Auction Mechanism for Robot Coordination* (2006).
[39] Vidal, J. M. and Durfee, E. H., The moving target function problem in multiagent learning, in *Proc. Third Int. Conf. on Multi-Agent Systems* (AAAI/MIT press, 1998), pp. 317–324.
[40] Whiteson, S., Taylor, M. E. and Stone, P., Empirical studies in action selection for reinforcement learning, *Adapt. Behav.* **15** (2007).
[41] Wolpert, D. H. and Tumer, K., Optimal payoff functions for members of collectives, *Adv. Complex Syst.* **4** (2001) 265–279.

Coevolution of Heterogeneous Multi-Robot Teams

Matt Knudson
Oregon State University
Corvallis, OR, 97331
knudsonm@engr.orst.edu

Kagan Tumer
Oregon State University
Corvallis, OR, 97331
kagan.tumer@oregonstate.edu

ABSTRACT

Evolving multiple robots so that each robot acting independently can contribute to the maximization of a system level objective presents significant scientific challenges. For example, evolving multiple robots to maximize aggregate information in exploration domains (e.g., planetary exploration, search and rescue) requires coordination, which in turn requires the careful design of the evaluation functions. Additionally, where communication among robots is expensive (e.g., limited power or computation), the coordination must be achieved passively, without robots explicitly informing others of their states/intended actions. Coevolving robots in these situations is a potential solution to producing coordinated behavior, where the robots are coupled through their evaluation functions. In this work, we investigate coevolution in three types of domains: (i) where precisely n homogeneous robots need to perform a task; (ii) where n is the optimal number of homogeneous robots for the task; and (iii) where n is the optimal number of *heterogeneous* robots for the task. Our results show that coevolving robots with evaluation functions that are locally aligned with the system evaluation significantly improve performance over robots evolving using the system evaluation function directly, particularly in dynamic environments.

Categories and Subject Descriptors

I.2.6 [**AI**]: Learning

General Terms

Algorithms, Experimentation

Keywords

Robot coordination; Coevolution; Team Formation

1. INTRODUCTION

Coordinating multiple robots to achieve a system-wide objective in an unknown and dynamic environment is critical to many of today's relevant applications, including the autonomous exploration of planetary surfaces and search and rescue in disaster response. In such cases, the environment may be dangerous, uninhabitable to humans all together, or sufficiently distant from central control that response times require autonomous, coordinated behavior. Evolutionary algorithms are particularly relevant to these applications, as solutions to robotic behavior in such complex environments are difficult or impossible to model.

In general, most multi-robot tasks can be broadly categorized into [8]: (i) tasks where a single robot can accomplish the task, but where having a multi-robot system improves the process (for example, terrain mapping or trash collection); and (ii) tasks where multiple robots are necessary to achieve a task (for example to carry an object). In both cases, coordination requires addressing many challenges (low level navigation, high level decision making, inter-robot coordination) each of which requires some degree of information gathering [17]. However, in the first case, a failure of coordination leads to inefficient use of resources, whereas in the second, it leads to a complete system breakdown. Therefore, a delicate balance must be established within a robots' behavior such that coordination is achieved without an overly strict adherence to a specific coordination protocol. Through coevolution, robots are given the freedom to develop their own protocols to benefit the system objective.

In this work, we focus on problems of the second type, and investigate the robot evaluation functions that need to be derived for the overall system to achieve high levels of performance. To that end, we investigate the use of difference evaluation functions to promote team formation [3]. Such evaluation functions have previously been applied to multiagent coordination problems of the first type [1, 18]. The key contribution of this work is to extend those results to coordination problems of the second type where unless tight coordination among the agents is established and maintained, the tasks cannot be accomplished. We develop teams within the multi-robot system using passive means (e.g., no explicit coordination directives) through the coupling of the robots' evaluation functions.

The application domain we selected is a distributed information gathering problem. First we explore the case where unless a particular point of interest is observed by n robots, the point of interest is not considered as observed. Second we explore the case where there is an optimal number of robots (n) that need to observe a point of interest, but where the system receives some value for observations by teams with other than n members. Finally, we construct a system where

GECCO'10, July 7–11, 2010, Portland, Oregon, USA.

the individuals are of differing capabilities, and one of each type is needed to provide optimal behavior.

In Section 2 we discuss the robot exploration problem. In Section 3, we present the problem requiring team formation. In Section 4 we present the problem of encouraging rather than requiring team formation, and in Section 5 we present heterogeneous teams with robots of two types. Finally in Section 6 we discuss the implication of these results and highlight future research directions.

1.1 Related Work

Extending single robot approaches to multi-robot systems presents difficulties in ensuring that the robots learn a particular task beneficial to the overall system. New approaches that are particularly well suited to multi-robot systems include using Markov Decision Processes for online mechanism design [15], developing new reinforcement learning based algorithms [4, 6, 9, 10], devising agent-specific evaluation functions [3], and domain based evolution [5]. In addition, forming coalitions for purposes of reducing search costs [11], employing multilevel learning architectures for the formation of coalitionsl [16], and market based approaches [21] have been examined.

The use of evolutionary algorithms in a multiagent domain is attractive due to the complex, non-Markovian nature of most systems. Coevolution furthers the advantages by evaluating the performance of individuals based on the interactions with others within the system. Coevolution algorithms tend to favor stability over optimality however [19], finding stable equilibria in agent behavior. One method used to alleviate this tendency is biasing the evaluation functions such that the fitness is evaluated on the most beneficial collaborative agents [13, 14]. The work in this paper is similar, where the most beneficial collaborators are those robots that most closely observe a Point of Interest, evaluated through a difference function. In addition, cooperative coevolution was further classified by defining a robustness criterion, demonstrated on a set of standard multiagent problems [20]. An interesting further extension to coevolution encodes individual agents with a base skill-set [7], preventing coevolved agents from having to learn the same thing independently.

2. ROBOT COORDINATION

The multi-robot information gathering problem we investigate in this work consists of a set of robots that must observe a set of points of interest (POIs) within a given time window [3]. The POIs have different importance to the system, and each observation of a POI yields a value inversely related to the distance the robot is from the POI. In addition, and particular to the work presented in this paper, multiple observations of a POI are either required (Section 3) or highly beneficial (Section 4) to the system objective.

2.1 Robot Capabilities

Each robot uses an evolutionary algorithm to map its sensor inputs to an x, y translation relative to the current position of the robot. Each robot utilizes a two layer sigmoid activated artificial neural network to perform this mapping.

The inputs to this neural network are four POI sensors (Equation 1) and four robot sensors (Equation 2), where x_q^{POI} and x_q^{ROBOT} provide the POI and robot "richness" of each quadrant q, respectively, V_j and L_j are the value and location of POI j respectively, L_i is the location of the current robot i and $\theta_{j,q}$ is the separation in radians between the POI and the center of the sensor quadrant.

$$x_{i,q}^{POI} = \sum_j \frac{V_j}{\delta(L_j, L_i)} \left(1 - \frac{|\theta_{j,q}|}{(\pi/4)}\right) \quad (1)$$

$$x_{i,q}^{ROBOT} = \sum_{k,k\neq i} \frac{1}{\delta(L_k, L_i)} \left(1 - \frac{|\theta_{k,q}|}{(\pi/4)}\right) \quad (2)$$

The two outputs indicate the velocity of the robot (in the two axes parallel and perpendicular to the current robot heading). The weights of the neural network are adjusted through an evolutionary search algorithm [3, 2] for ranking and subsequently locating successful networks within a population [12, 3]. The algorithm maintains a population of ten networks, utilizes mutation to modify individuals, and ranks them based on a performance metric specific to the domain. The search algorithm used is shown in Figure 1 which displays the ranking and mutation steps.

Initialize N networks at $T = 0$
For $T < T_{max}$ Loop:

1. Pick a random network N_i from population

   With probability ϵ: $N_{current} \leftarrow N_i$

   With probability $1 - \epsilon$: $N_{current} \leftarrow N_{best}$
2. Mutate $N_{current}$ to produce N'
3. Control robot with N' for next episode
4. Rank N' based on performance (evaluation function)
5. Replace N_{worst} with N'

Figure 1: Evolutionary Algorithm: An ϵ-greedy evolutionary algorithm to determine the weights of the neural networks. See text body for definitions. T indexes episodes, N indexes networks with appropriate subscripts, and N' is the modified network for use in control of the current episode.

In this domain, mutation (Step 2) involves adding a randomly generated number to every weight within the network. This can be done in a large variety of ways, however it is done here by sampling from a random Cauchy distribution where the samples are limited to the continuous range [-10.0,10.0] [3]. Ranking of the network performance (Step 4) is done using a domain specific evaluation function, and is discussed in the following section.

2.2 Robot Objectives

In these experiments, we used three different evaluation functions [3] to determine the performance of the robot: the system evaluation function which rates the performance of the full system; a local evaluation function that rates the performance of a "selfish" robot; and a difference evaluation function that aims to capture the impact of a robot in the multi-robot system [3]. These three evaluation functions are:

- The system evaluation reflects the performance of the full system. Though robots optimizing this evaluation function guarantees that the robots all work toward

the same purpose, robots have a difficult time discerning their impact on this function, particularly as the number of robots in the system increases.

- The local evaluation reflects the performance of the robot operating alone in the environment. Each robot is rewarded for the sum of the POIs it alone observed. If the robots operate independently, optimizing this evaluation function would lead to good system behavior. However, if the robots interact frequently, then each robot aiming to optimize its own local function may lead to competitive rather than cooperative behavior.

- The difference evaluation reflects the impact a robot has on the full system [3, 2]. By removing the value of the system evaluation where robot i is inactive, the difference evaluation computes the value added by the observations of robot i alone. Because only POIs to which robot i were closest need this difference computed, this evaluation function is "locally" computable in most instances.

Though conceptually the same, the specifics of these evaluations are different for each of the problems described in the following sections. We derive those specific evaluation structures and present the experimental results below.

3. REQUIRING TEAM FORMATION

In the first problem we examine, the robots need to form teams to perform a task and contribute to the system objective. In this problem, a POI is considered observed only if n robots visit that POI from within a certain observation distance. Neither the robot, nor the system receive any value unless multiple observations of a POI occur. This problem formulation ensures that the problem is one that cannot be solved by a single robot and that the team formation is essential to the completion of each task.

3.1 Problem Definition

To formalize this problem, let us first focus on a problem where the observations of the two robots closest to a POI are tallied. If more than two robots visit a POI, only the observations of the closest two are considered and their visit distances are averaged in the computation of the system evaluation (G), which is given by:

$$G(z) = \sum_i \sum_j \sum_k \frac{V_i \; N^1_{i,j} \; N^2_{i,k}}{\frac{1}{2}(\delta_{i,j} + \delta_{i,k})} \tag{3}$$

where V_i is the value of the ith POI, $\delta_{i,j}$ is the closest distance between jth robot and the ith POI, and $N^1_{i,j}$ and $N^2_{i,k}$ determine whether a robot was within the observation distance δ_o and the closest or second closest robot, respectively, to the ith POI:

$$N^1_{i,j} = \begin{cases} 1 & \text{if } \delta_{i,j} < \delta_o \text{ and } \delta_{i,j} < \delta_{i,l} \; \forall l \neq j \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

and

$$N^2_{i,k} = \begin{cases} 1 & \text{if } \delta_{i,k} < \delta_o \text{ and } \delta_{i,k} < \delta_{i,l} \; \forall l \neq j,k \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

The single robot evaluation function used by each robot only focuses on the value a robot receives for observing a particular POI, and results in:

$$P_j(z) = \sum_i \frac{V_i}{\delta_{i,j}} \qquad \text{if } \delta_{i,j} < \delta_o \tag{6}$$

This evaluation promotes selfish behavior only, providing a clear, easy-to-learn signal, but one not aligned with the system objective as a whole.

Finally, the difference evaluation for a robot aims to provide system-wide beneficial behavior, while remaining sensitive to the actions of a robot [3]. This difference evaluation function is given by:

$$D_j(z) = \begin{cases} \sum_i \left(\frac{V_i}{\frac{1}{2}(\delta_{i,j}+\delta_{i,k})} - \frac{V_i}{\frac{1}{2}(\delta_{i,j}+\delta_{i,l})} \right) & \text{if } \delta_{i,j}, \delta_{i,k} < \delta_{i,l} < \delta_o \\ \sum_i \frac{V_i}{\frac{1}{2}(\delta_{i,j}+\delta_{i,k})} & \text{if } \delta_{i,j}, \delta_{i,k} < \delta_o \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

where l is the third closest robot to POI i (meaning that robots j and k are the closest two for the first two conditionals). All three of these evaluations were applied for learning in many different situations, though for brevity, only an environment with 50 POIs and 40 robots (which was representative of the general performance of the evaluations) is presented.

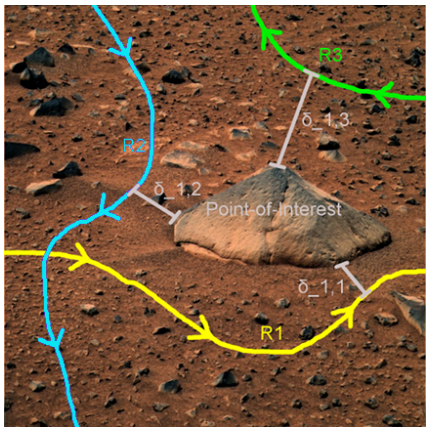

Figure 2: Sample robot paths in an exploration scenario. Multiple observations are made of a particular point of interest. In the team formation domain, multiple observations must be made for the POI to have any value to the system. Background courtesy of JPL.

Figure 2 shows a schematic of how these evaluation functions are computed, given that all three robots are within the observation radius. Only robots 1 and 2 ($R1$ and $R2$) are taken into consideration when calculating $G(z)$ because their observation distance ($\delta_{1,1}$ and $\delta_{1,2}$) are closer than $R3$ ($\delta_{1,3}$). For $G(z)$, robot 3's observation is discarded. For the difference evaluation for robots 1 or 2, robot 3 is taken into consideration. For example, in calculating Equation 7 for $R2$, the first term considers $R1$ and $R2$, where the second term considers $R1$ and $R3$. That is, $R2$ receives the difference between the observation values of $R1$ and $R2$ and the observation values of R1 and R3.

3.2 Results

The environment used for presentation in this paper contained 40 robots and 50 POIs, providing a great deal of

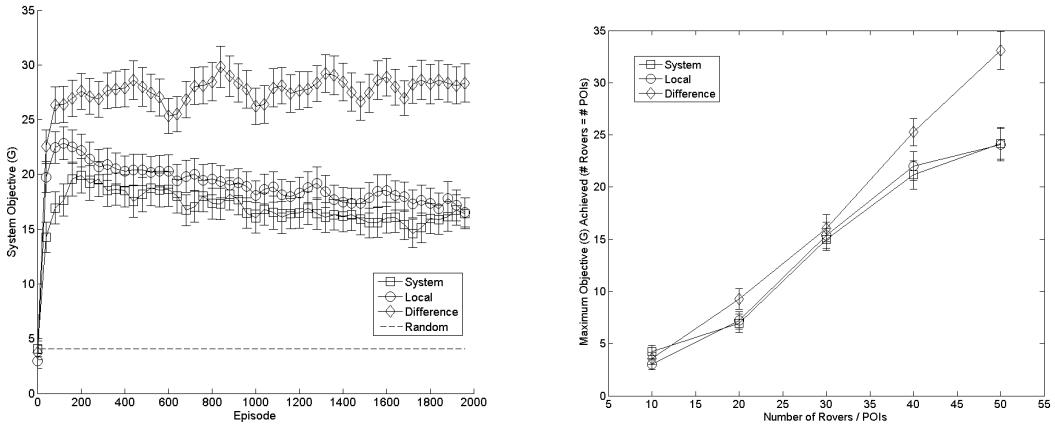

Figure 3: *Team Formation Required* **Left: System evaluation is plotted versus episode for learning in an environment containing 40 robots and 50 POIs. Right: Maximum evaluation achieved is plotted for equal numbers of robots and POIs. Learning is done with system, local, and difference evaluations requiring the formation of teams of two robots.**

information to be gathered, while simultaneously creating a congested situation. In addition, the environment was highly dynamic, where 10% of the POIs (selected randomly) changed location and value at each episode. This was done to encourage specific coordination behavior based on sensor inputs rather than specific x-y coordinates. The results are based on 2000 episodes of 30 time-steps each, and are averaged for significance.

Figure 3 (*left*) shows that robots using all three evaluations perform significantly better than random behavior. It also shows that the difference evaluation provides a signal that allows the robots to learn to coordinate their actions, whereas using the system and local evaluations do not. Additionally, Figure 3 (*right*) shows that the difference evaluation does not provide benefits until the system reaches the point of high complexity.

4. ENCOURAGING TEAM FORMATION

In the second problem we examine, multiple robots are encouraged (rather than required) to form teams to perform a task and contribute to the system objective. In this problem, a POIs value is optimized for n robots observing it, but the system receives lesser value for other numbers of robots observing the POI. Figure 4 shows the functional form of the two system evaluations used in Section 3 and Section 4.

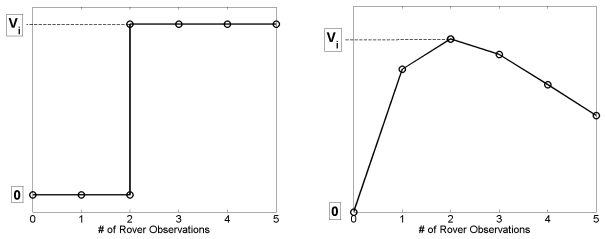

Figure 4: **POI value structure is compared between the required (left) and encouraged (right) team formation systems.**

4.1 Problem Definition

For these evaluations, δ_o remains the same, however the distance of observation is no longer explicitly included in the evaluation function, relying on inherent inclusion in the observation radius of the POI. As before, three evaluation functions are defined, beginning with the system evaluation given by:

$$G(z) = \sum_i \alpha V_i x e^{\frac{-x}{\beta}} \tag{8}$$

where i indexes POIs, x is the number of robots within δ_o, β is the observation capacity, and α is a constant chosen to be 1.37 such that the maximum of the exponential curve approximates the POI value V_i.

For this new system evaluation, the selfish robot evaluation is defined as:

$$P_j(z) = \sum_{i_j} \alpha V_{i,j} x e^{\frac{-x}{\beta}} \tag{9}$$

where indexing and constant selection is the same as above. This evaluation includes no information regarding contribution to the system as a whole, rather indicating only what robot j can directly observe. This robot evaluation is the component of the system objective for which robot j was within the observation distance δ_o of each POI.

Finally, the difference evaluation function for this system results in:

$$D_j(z) = \sum_{i_j} \alpha V_{i,j} \left[x e^{\frac{-x}{\beta}} - (x-1)\, e^{\frac{-(x-1)}{\beta}} \right] \tag{10}$$

where indexing and constant selection is the same as above. This evaluation aims to provide the contribution of robot j to the system. The performance of all three evaluation functions are presented in the next section.

4.2 Results

All training parameters were maintained from those used in Section 3.2, including the number of POIs and robots.

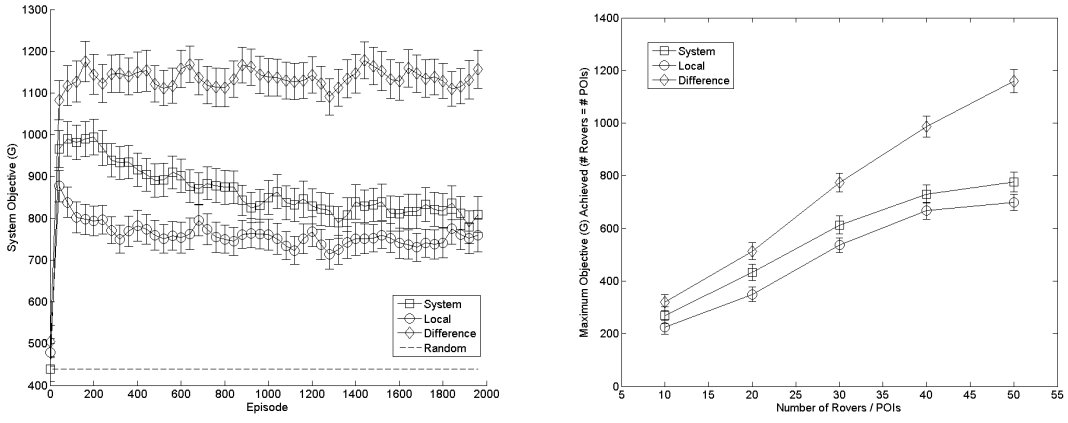

Figure 5: ***Team Formation Encouraged*** **Left: System evaluation is plotted versus episode for learning in an environment containing 40 robots and 50 POIs. Right: Maximum evaluation achieved is plotted for equal numbers of robots and POIs. Learning is done with system, local, and difference evaluation functions requiring the formation of teams of two robots.**

The results presented in Figure 5 are qualitatively similar to those seen in Figure 3. This is a good result, demonstrating that the team requirement in general is applicable and successful for multiple formulations of the problem (does not depend on the exact form of G). As before, the difference evaluation provides consistent behavior throughout, where the system evaluation function (aligned with system, but not sensitive to a given robot's actions) and local evaluation (sensitive to a robot's action, but not necessarily aligned with the system evaluation) break down.

Here again Figure 5 (*right*) shows that as the system increases in complexity, the difference evaluation, through providing a better learning signal, provides consistent behavior through the increased complexity of the system. The system and local learning evaluation function performance tapers off, where using the difference evaluation maintains its' performance slope, clearly indicating that when the number of robots within the system becomes large, the difference evaluation is able to maintain successful dynamic team formation. In addition, through encouraging team formation, rather than requiring it, we have presented a simpler problem to learn.

4.3 Higher Coordination Requirements

The previous two sections investigated coordination for $n = 2$, for both required and encouraged team formation scenarios. The behavior of the three evaluation functions was similar for both cases. In this section we investigate the behavior for $n = 3$, a change that has significant impact on the computation of G, particularly when the observation distance is not increased.

Figure 6 (*left*) shows the learning results for requiring three robots to observe a POI. The all-or-nothing learning structure in this evaluation function makes it very difficult for a robot using passive team formation to extract the relevant signal. This brings the difference evaluation closer to the system objective by reducing its sensitivity to a particular robot's actions (that is, in most cases, removing a robot from the system has no impact on the system performance). As a consequence, the difference evaluation fails to promote good system-level behavior.

By contrast, Figure 6 (*right*) shows the behavior of the system where team formation is encouraged by a decaying value assignment to POI observations. In this case, moving from $n = 2$ to $n = 3$ does not affect the difference evaluation. This is because in this problem, removing a robot has a computable impact on the system objective. This creates a "gradient" for evaluating the impact of a robot on the system as a whole. As a consequence, the difference evaluation performs better than system or local evaluation functions.

We combine the conclusions that a) encouraging dynamic teams, rather than requiring them, is more robust to changes in system definition and, b) difference evaluations are more successful in systems changing in the number of robots and POIs from the above sections to formulate a problem for heterogeneous team formation in the following section.

5. HETEROGENOUS TEAM FORMATION

The success in team formation shown in the above sections points to an investigation of teams constructed of heterogenous robots. When the entire team is made of robots of identical construction, the tasks are limited to general redundant observations of an environment to provide robustness, or mechanical tasks that require multiple individuals to provide enough effort. In contrast, if the individuals can learn to dynamically partner with one-another, the question arises whether or not, given additional sensing, individuals of differing construction can partner to provide a more specific suite of tasks.

5.1 Problem Definition

In the final problem we investigate, we define two robot types; *blue* and *green*. These can represent any number of possible construction differences, including sensing and articulation, depending on the system in which they are installed. The individuals must have the ability to determine the difference between the two, for example a blue robot must be able to determine that there are green robots elsewhere in the environment. In addition, the evaluation function must again be modified to represent the need for robots of differing capabilities to visit a POI.

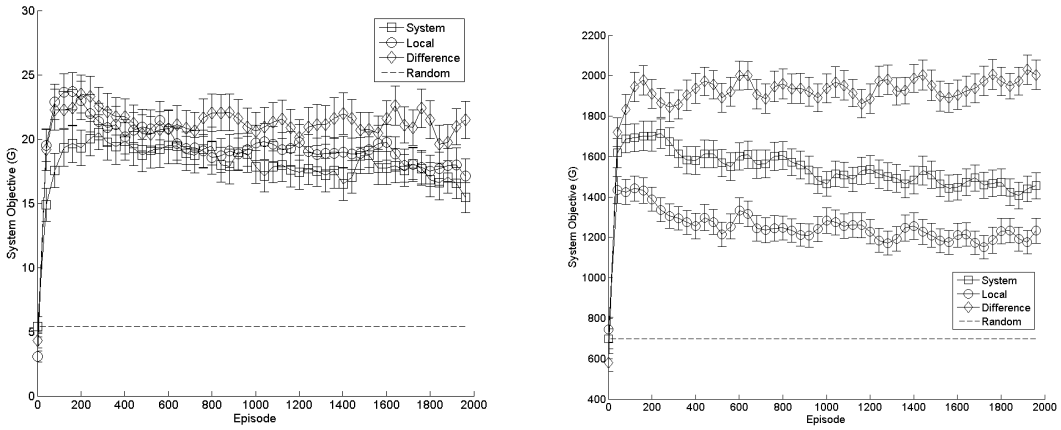

Figure 6: ***Higher Coordination Requirements*** *($n = 3$)* **Left: Required Team Formation. Right: Encouraged Team Formation. System evaluation is plotted versus episode for learning in an environment containing 40 robots and 50 POIs. Learning is done with system, local, and difference evaluation functions for three robots to observe a POI.**

The sensing capabilities are similar to those shown in Section 2.1. For each quadrant q however, the robot sensor is split into two, one indicating the density of "blue" robots and the other indicating "green" robots. This increases the number of inputs to the neural network from 8 to 12, and the number of hidden units was increased accordingly. This configuration maintains comparability to homogeneous applications while providing the differentiation between robot types needed by the new problem.

We showed that encouraging team formation is more beneficial to the learning process over requiring team formation, and therefore the modified evaluation function reflects the exponential form as much as possible. Again, δ_o remains the same, and the functional form includes the number of robots in the observation radius of a given POI. The number of observations however is separated into the number of blue robots and green robots that made observations. Therefore, the optimal solution is not only that two robots visit, but that one of each type visits each POI.

As with previous work, three evaluation functions were defined for comparison, reflecting the styles discussed in Section 2.2. Beginning with the system-level evaluation:

$$G(z) = \sum_i \alpha V_i x_b x_g e^{\frac{-x_b x_g}{\beta_b \beta_g}} \tag{11}$$

where x_{type} is the number of observations of a POI i of each type of robot, α is a scaling constant to ensure the maximum of the function approximates the POI value V_i (set to 2.72 for these experiments), and β_x are the constants to produce functional peaks at the desired number of observations of each type of robot. For example, to have one of each type observe a POI, $\beta_b = \beta_g = 1$, which is the configuration for subsequent experiments.

The local evaluation is similar to the above, however it reflects only the POIs that robot j has visited. Therefore it is locally computable and easy to learn, but does not indicate the robot's impact on the system as a whole:

$$P_j(z) = \sum_{i_j} \alpha V_{i,j} x_b x_g e^{\frac{-x_b x_g}{\beta_b \beta_g}} \tag{12}$$

where indexing and constant selection is the same as the above.

Finally, the difference evaluation includes information contained in the system-level evaluation, but is easier to learn as it directly indicates how robot j contributed to the system as a whole. It is contingent on the type of robot j:

$$D_j(z) = \sum_{i_j} \alpha V_{i,j} \left(x_b x_g e^{\frac{-x_b x_g}{\beta_b \beta_g}} - (x_b - 1) x_g e^{\frac{-(x_b-1) x_g}{\beta_b \beta_g}} \right) \tag{13}$$

where indexing and constant selection is the same as above. The equation shown is for robot j of type *blue*, where if the type is *green*, 1 is subtracted from the *green* robot observations rather than the *blue*. The experimental results for the use of all three evaluation functions follows in the next section.

5.2 Results

The domain for the experiments involving heterogeneous teams is the same as that used in the above work. Each robot is randomly assigned a type at the beginning of each experiment based on a given team ratio. Learning time is adjusted from 2000 episodes to 3000 as the network has increased in size, and the problem has increased in difficulty, slightly decreasing convergence speed. The environment maintains its dynamic nature, where 10% of the POIs change location and value at every episode, though the robots maintain their type throughout the learning process.

Figure 7 (*left*) shows the results of training in an environment where 40 robots and 50 POIs are present. The ratio of *blue* to *green* robots is 50%, meaning there are 20 of each type present. With the increased problem complexity we observe that the local evaluation is entirely incapable of learning a good solution, in fact learning the wrong thing, performing worse than random parameter selection (network weights) after convergence.

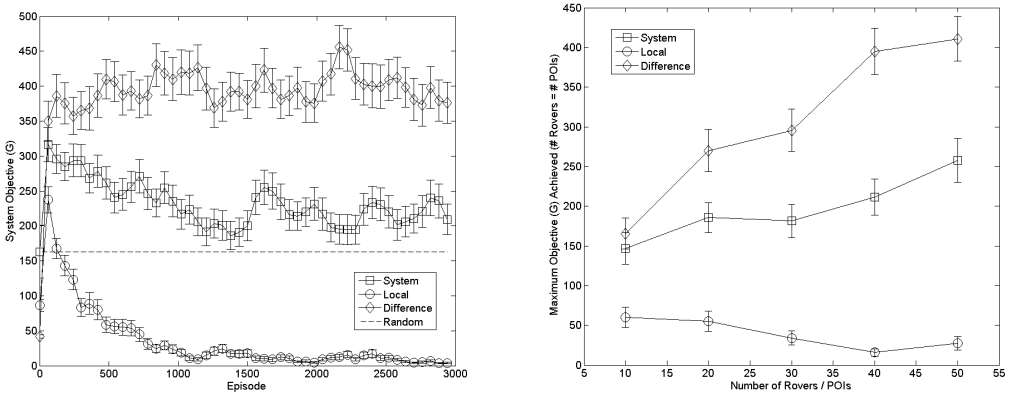

Figure 7: ***Heterogeneous Team Formation*** **Left: System performance for an environment containing 40 robots and 50 POIs. Learning is done with system, local, and difference evaluation functions requiring the formation of teams of two robots, one of each type. Right: Maximum performance achieved for equal numbers of robots and POIs. Learning is done with system, local, and difference evaluation functions encouraging the heterogeneous formation of teams of two robots.**

As with the results in Section 4.2, learning with the system-level evaluation function proves difficult, as there is a great deal of information contained in the signal; too much regarding other robots for each individual to ascertain what actions are best in contributing to the system as a whole. The difference evaluation however, as expected, learns quickly and maintains performance through the learning process. This confirms the applicability of the difference evaluation in general, and specifically indicates that dynamically requiring heterogeneous team formation in a congested and dynamically changing environment is achievable, indeed successful.

We next examine the impact of increasing both the number of robots and the number of POIs within the system simultaneously. Figure 7 (*right*) shows the maximum system-level evaluation function achieved for varying numbers of robots and POIs (where the number of robots and POIs is the same). The local evaluation begins poorly and decreases further as the system complexity increases, as shown in previous figures. Using the system-level evaluation for learning, while increasing slightly as complexity increases, is strongly outperformed by the difference evaluation. As with all previous dynamic team formation work in this paper, utilization of the evaluation function significantly improves performance over the others, and provides an excellent learning signal for dynamic team formation, particularly in domains absent of communication and heterogeneous in construction.

In varying the ratio between robot types present in the system, we can determine if the robots are able to modify their behavior to suit changes in system consistently. For example, if a large set of robots of a specific type fail, the system must have the ability to adjust coordination behavior to maintain success in accomplishing the tasks requested. Figure 8 shows the maximum system performance achieved when the ratio between *blue* and *green* robots is varied. The variance is symmetrical, therefore 10% *blue* and 90% *green* is the same as 10% *green* and 90% *blue*. The number of robots and POIs present in the system is held constant.

The local evaluation always performs poorly, and the ratio of types within the system has little impact on the performance of the system evaluation. This points to a lack of attention paid to the heterogeneous nature of the team in the behavior of the robots that learn with the system evaluation. The difference evaluation however varies significantly when the teams are strongly unbalanced, particularly when the ratio is set to 20%. This is due to the variance in sensing information during the learning process. For example, when there are much fewer robots of one type within the system, the sensors detecting the two types return significantly different levels of information, and therefore the algorithm can learn to focus on the sensors showing where robots of a different type are located. This provides additional information to the algorithm regarding the actions that will lead directly to an increase in the learning evaluation performance.

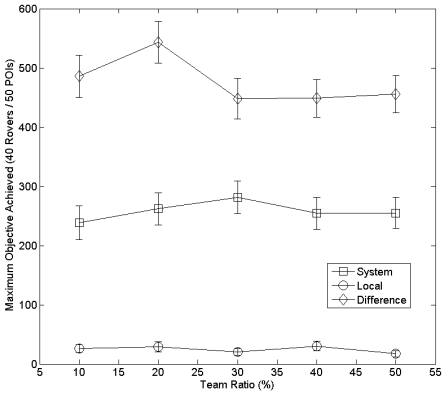

Figure 8: **Heterogeneous Team Ratios: System evaluation is plotted versus episode for learning in an environment containing 40 robots and 50 POIs. Learning is done with system, local, and difference evaluation functions requiring the formation of teams of two robots, one of each type. The ratio between** $blue$ **and** $green$ **robots varies in the system.**

6. DISCUSSION AND FUTURE WORK

Exploration of planetary surfaces or in disaster response requires that robotic solutions operate in unknown and dynamic environments. Coordinating multiple robots in such domains presents additional challenges. In this work, we explore multi-robot coordination domains where multiple robots are necessary to achieve a task (for example to carry an object). We focus on passive coordination that is accomplished through the robots' evaluation functions.

The work presented is this paper explores three types of problems where robot coordination is beneficial. First, we explore a problem where n robots must coordinate to receive a reward. Then, we explore a problem where the system reward is optimized for n robots, but other number of robots observing a POI also contribute to the system objective. Finally we develop a heterogeneous system where two types of robots are present, and an observation by one of each produces optimal behavior.

In all three cases, coordination and team formation is established and maintained through passive means encoded in the robots evaluation functions. The difference evaluation yielded the best results because it provided an evaluation that was aligned with the overall system evaluation, while maintaining sensitivity to a robot's actions, even when many robots were active within the coordinated system. That approach also extended to three or more robots encouraged to complete a task. This is an interesting result showing that the difference evaluation is best suited to domains where the impact of a robot on a system can be ascertained.

We are currently implementing the work discussed in this paper in robot hardware. This involves investigating non-episodic learning such that coordination and ad-hoc team formation can be learned while the robot is in current operation. In addition, extensions to the learning algorithm used in this paper will be investigated to facilitate the restrictions of physical hardware.

Acknowledgments

This work was partially supported by AFOSR grant FA9550-08-1-0187 and NSF grant IIS-0910358.

7. REFERENCES

[1] A. Agogino and K. Tumer. Distributed evaluation functions for fault tolerant multi rover systems. In *Proceedings of the Genetic and Evolutionary Computation Conference*, Seattle, WA, July 2006.

[2] A. K. Agogino and K. Tumer. Analyzing and visualizing multiagent rewards in dynamic and stochastic environments. *Journal of Autonomous Agents and Multi Agent Systems*, 17(2):320–338, 2008.

[3] A. K. Agogino and K. Tumer. Efficient evaluation functions for evolving coordination. *Evolutionary Computation*, 16(2):257–288, 2008.

[4] M. Ahmadi and P. Stone. A multi-robot system for continuous area sweeping tasks. In *Proceedings of the IEEE Conference on Robotics and Automation*, pages 1724–1729, May 2006.

[5] M. Alden, A.-J. van Kesteren, and R. Miikkulainen. Eugenic evolution utilizing a domain model. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2002)*, San Francisco, CA, 2002.

[6] C. Claus and C. Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. In *Proceedings of the Artificial Intelligence Conference*, pages 746–752, Madison, WI, July 1998.

[7] D. B. D'Ambrosio and K. O. Stanley. Generative encoding for multiagent learning. In *Genetic and Evolutionary Computation Conference*, 2008.

[8] B. P. Gerkey and M. J. Mataric. Multi-robot task allocation: Analyzing the complexity and optimality of key architectures. In *Proceedings of the IEEE Int. Conference on Robotics and Automation*, pages 3862–3868, 2003.

[9] C. Guestrin, M. Lagoudakis, and R. Parr. Coordinated reinforcement learning. In *Proceedings of the 19th International Conference on Machine Learning*, page 41Ű48, 2002.

[10] J. Hu and M. P. Wellman. Multiagent reinforcement learning: Theoretical framework and an algorithm. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 242–250, 1998.

[11] E. Manisterski, D. Sarne, and S. Kraus. Enhancing mas cooperative search through coalition partitioning. In *Proc. Int'l Joint Conference on Artificial Intelligence*, pages 1415–1421, 2007.

[12] S. Nolfi, D. Floreano, O. Miglino, and F. Mondada. How to evolve autonomous robots: Different approaches in evolutionary robotics. In *Proc. of Artificial Life IV*, pages 190–197, 1994.

[13] L. Panait. Improving coevolutionary search for optimal multiagent behaviors. In *International Joint Conference on Artificial Intelligence*, pages 653–658. Morgan Kaufmann, 2003.

[14] L. Panait, S. Luke, and R. P. Wiegand. Biasing coevolutionary search for optimal multiagent behaviors. *IEEE Transactions on Evolutionary Computation*, 10(6):629–645, 2006.

[15] D. Parkes and S. Singh. An MDP-based approach to online mechanism design. In *NIPS 16*, pages 791–798, 2004.

[16] L. Soh and X. Li. An integrated multilevel learning approach to multiagent coalition formation. In *Proc. Int'l Joint Conference on Artificial Intelligence*, pages 619–625, 2003.

[17] S. Thrun and G. Sukhatme. *Robotics: Science and Systems I*. MIT Press, 2005.

[18] K. Tumer and A. Agogino. Coordinating multi-rover systems: Evaluation functions for dynamic and noisy environments. In *The Genetic and Evolutionary Computation Conference*, 2005.

[19] R. P. Wiegand, W. Liles, and K. D. Jong. *Modeling variation in cooperative coevolution using evolutionary game theory*, pages 203–220. Morgan Kaufmann, 2002.

[20] R. P. Wiegand and M. A. Potter. Robustness in cooperative coevolution. In *Genetic and Evolutionary Computation Conference*, pages 369–376. ACM Press, 2006.

[21] Y. Ye and Y. Tu. Dynamics of coalition formation in combinatorial trading. In *Proc. Int'l Joint Conference on Artificial Intelligence*, pages 625–632, 2003.